NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

DEVELOPMENT OF A LONG-RANGE GLIDING UNMANNED UNDERWATER VEHICLE UTILIZING JAVA SUN SPOT TECHNOLOGY

by

Ronald J. Hemmelgarn

September 2008

Thesis Advisor:	Don Brutzman
Second Reader:	Jeff Weekley

This thesis was done at the MOVES Institute

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	September 2008	Master's Thesis

4. TITLE AND SUBTITLE: Development of a Long-Range Gliding Underwater Vehicle Utilizing Java Sun SPOT Technology	**5. FUNDING NUMBERS**
6. AUTHOR(S) Ronald J. Hemmelgarn	
7. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME AND ADDRESS N/A	**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT	**12b. DISTRIBUTION CODE**
Approved for public release; distribution is unlimited	

13. ABSTRACT (maximum 200 words)

The future of U.S. Naval Operations can be described by a simple system of requirements and constraints. Increasing the diversity and scope of mission requirements, while being constrained by decreasing budget resources, requires some form of equalization to maintain a constant rate of successful mission fulfillment. The solution to this system can be found in unmanned vehicle development. The most recent revision of the Navy Unmanned Undersea Vehicle (UUV) Master Plan outlined the need to develop a cost-effective, flexible program by maximizing modularity and commonality of UUVs. This thesis investigates the convergence of three main areas of UUV development; mission flexibility, modular control systems, and hardware in-the-loop testing and analysis. This work also evaluates the feasibility of a potential solution to support those objectives. Hardware-in-the-loop simulation and testing of embedded systems is a proven method for effectively testing complex systems, helping to reduce the risks of developing or deploying an ineffective costly system. An innovative glider design by the University of Toulon, France is the subject of this study. Unlike most rigid-hull gliders, the scalable free-flood volume of this vehicle holds the promise of carrying significant payload as long as overall buoyancy remains neutral. The research and development described in this thesis utilizes an existing planning and simulation tool, combined with an improved low-cost embedded-system robot controller, to test and evaluate a new free-flood, long-range gliding underwater vehicle. This proposed solution utilizes both open-source hardware and software solutions to design a prototype gliding underwater vehicle. Further work is needed to demonstrate the efficiency and effectiveness of this design.

14. SUBJECT TERMS Java, UUV, AUVW, Sun SPOT, Glider, Autonomous Underwater Vehicle Workbench, Unmanned Underwater Vehicle	**15. NUMBER OF PAGES** 117
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89) Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

DEVELOPMENT OF A LONG-RANGE GLIDING UNMANNED UNDERWATER VEHICLE UTILIZING JAVA SUN SPOT TECHNOLOGY

Ronald J. Hemmelgarn
Lieutenant, United States Navy
B.S., Old Dominion University, 2003

Submitted in partial fulfillment of the
requirements for the degree of

**MASTER OF SCIENCE IN
MODELING, VIRTUAL ENVIRONMENTS AND SIMULATION (MOVES)**

from the

**NAVAL POSTGRADUATE SCHOOL
September 2008**

Author: Ronald J. Hemmelgarn

Approved by: Don Brutzman, Ph.D.
Thesis Advisor

Jeff Weekley
Second Reader

Mathias Kölsch
Chair, MOVES Academic Committee

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The future of U.S. Naval Operations can be described by a simple system of requirements and constraints. Increasing the diversity and scope of mission requirements, while being constrained by decreasing budget resources, requires some form of equalization to maintain a constant rate of successful mission fulfillment. The solution to this system can be found in unmanned vehicle development. The most recent revision of the Navy Unmanned Undersea Vehicle (UUV) Master Plan outlined the need to develop a cost-effective, flexible program by maximizing modularity and commonality of UUVs. This thesis investigates the convergence of three main areas of UUV development; mission flexibility, modular control systems, and hardware in-the-loop testing and analysis. This work also evaluates the feasibility of a potential solution to support those objectives.

Hardware-in-the-loop simulation and testing of embedded systems is a proven method for effectively testing complex systems, helping to reduce the risks of developing or deploying an ineffective costly system. An innovative glider design by the University of Toulon, France is the subject of this study. Unlike most rigid-hull gliders, the scalable free-flood volume of this vehicle holds the promise of carrying significant payload as long as overall buoyancy remains neutral. The research and development described in this thesis utilizes an existing planning and simulation tool, combined with an improved low-cost embedded-system robot controller, to test and evaluate a new free-flood, long-range gliding underwater vehicle. This proposed solution utilizes both open-source hardware and software solutions to design a prototype gliding underwater vehicle. Further work is needed to demonstrate the efficiency and effectiveness of this design.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

LIST OF ACRONYMS AND ABBREVIATIONS

ALAV	Autonomous Light Air Vessel
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
AUSI	Autonomous Undersea Systems Institute
AUV	Autonomous Underwater Vehicle
AUVW	Autonomous Underwater Vehicle Workbench
CPU	Central Processing Unit
FCC	Federal Communications Commission
GPIO	General Purpose Input Output
GPS	Global Positioning System
HTML	Hyper Text Markup Language
ISITV	Institute of Engineering and Science of Toulon, France
JVM	Java Virtual Machine
LED	Light Emitting Diode
MMCX	Micro Mate Connector
NPS	Naval Postgraduate School
SAUV	Solar-powered Autonomous Underwater Vehicle
SDK	Software Development Kit
SPOT	Small Programmable Object Technology
TSI	Technology Systems Incorporated
UART	Universal Asynchronous Receiver/Transmitter

UUV	Unmanned Underwater Vehicle
XML	Extensible Markup Language
XSLT	Extensible Stylesheet Language for Transformations

ACKNOWLEDGMENTS

As with any great endeavor, this thesis would not have been possible without the contributions of many, both professional and personal. I would like to acknowledge the support and professional dedication of the following:

-My thesis advisor Don Brutzman, Second Reader Jeff Weekley, and members of the SAVAGE research group for all the guidance and assistance.

-The design and development by Professor Didier Leandri, and all of his technical support, as well as the construction and testing of the first-generation vehicle by his laboratory at the University of Toulon in France.

-The prior thesis study by the team of French students visiting NPS which served as the basis for this thesis.

-The assistance and support from the team at Sun Microsystems, as well as the entire Sun SPOT community.

-The support from the NPS Center for AUV Research.

I would also like to take this opportunity to pass on my deepest heart-felt personal thank you to the following for their unrelenting support:

-Mike Cornwell, my brother from another mother. Thank you for being there for me, through thick and thin, and for not offering any advice. I can only hope for the opportunity to be the friend you have been to me.

-Rob "Fatty B" Betts, you have been a true friend and comrade. My entire experience at NPS, both personally and professionally, has only been made better by your involvement. I will now also acknowledge that you did in fact tell me so.

-My extended family at Carbone's. Thank you Sal, Nick, Missy, Anna, Lorraine, and of course Lindsay for your undying support, friendship and for always putting things back in perspective at the end of the day. Each and every one of you deserves a pile of credit for getting this across the finish line.

-I would like to acknowledge that if you are in fact going to be dumb, you had better be tough.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. OVERVIEW

This thesis is a continuation of ongoing research at the University of Toulon France and the Naval Postgraduate School into a notional long-range unmanned underwater gliding vehicle previously published under the name SeaDiver. This type of gliding vehicle has several characteristics inherent in its design that set it apart from other gliding unman underwater vehicles (UUVs). Use of a wing cross-section as the hull form permits construction of larger vehicles which might carry significantly larger science payloads if neutral buoyancy is maintained.

Gliding UUVs are on the forefront of research in the UUV field. Their ability to perform long-range missions efficiently to collect oceanographic data from a much larger sample area than is possible from a manned research vessel is what makes such vehicles so desirable. However, the primary limitation for most gliding UUVs is inherent in the hull design. Most have rigid pressurized hulls, with a fixed volume and limited payload capability. The payload usually includes oceanographic sensors for data collection, and the vehicle is designed around a specific payload or mission. As the size and payload requirements change, so must the vehicle size, and therefore many of the vehicle dynamics change as well. Such changes often require a completely new vehicle design, making most glider designs inflexible and tightly optimized to a single mission.

Gliding UUVs are characterized by four common features:

- Buoyancy driven propulsion systems that use volume change to create vertical lift, and some sort of wing shape to convert that to forward motion.
- Sawtooth pattern of vertical motion.
- Relatively slow speeds.
- Long duration due to low power consumption for propulsion.

The initial appeal of this vehicle type (besides the long duration of operation) is the sawtooth-vertical motion pattern. This type of pattern is excellent for collecting ocean profile information, as it can sample the entire range of desired depths. (Davis, Eriksen, & Jones, 2003)

The majority of gliding vehicle designs being tested today adopt the cylindrical body type that houses the buoyancy system, often adding long narrow wings to convert depth changes into forward motion. This thesis reports on research that aims to prove there is a new vehicle design that can be scaled up or down in size, carry much larger payloads than typical gliding vehicles, and accomplish this wider range of missions economically.

B. PROBLEM STATEMENT

The design of a low-cost gliding unmanned underwater vehicle was outlined in the technical document by Gassier et al., "Implementing a Low-Cost Long-Range Unmanned Underwater Vehicle: the SeaDiver Glider" conducted at NPS. (Dumonteil, Gassier, & Rebollo, 2006) The implementation left the challenge of building an electrically economical, suitably powerful control system to support long-range tactical operations as future work. Sun Microsystems has recently released the Java SunSPOT, a versatile Java-based microcontroller that presents itself as a viable option to suitably control the gliding underwater vehicle. SunSPOT control might also provide the needed interfaces and functionality to integrate the physical vehicle in the simulation, testing, and analysis process.

C. MOTIVATION

An extensive amount of research has gone into designing the notional SeaDiver gliding vehicle, as well as designing and running simulations to assess its ability to perform tactical missions. However, the simulations that were created and populated with notional parameters and test results were inconclusive due to lack of actual test results from a physical model. Motivations for this research are to create the physical model, to build a suitable control system to close the loop in testing and evaluation of this notional design, and to evaluate the Sun Microsystems SunSPOTs suitability and effectiveness as an economic vehicle-control solution.

This research addresses the following questions:

- Can Java SunSPOT technology be effectively employed to control and support long-range glider operation?

- Does the control system support expandability in communication and guidance to support open-ocean employment?
- Is the long-range glider an economical option for employment in open-ocean situations?

D. OBJECTIVES

This thesis focuses on developing and evaluating a suitable physical model of a notional gliding vehicle, and building the interfaces necessary to directly link to existing open-source NPS simulation software, in order to capitalize on the benefits of the use of simulation in the development of unmanned vehicles. The primary objective of this research is to evaluate an emerging micro-controller technology, the Sun SPOT from Sun Microsystems, assessing its ability to control a prototype gliding underwater vehicle, as well as to provide the functionality needed to interface with existing simulation software to achieve true physical model in-the-loop testing and analysis capabilities for future vehicle designs.

The innovative vehicle design is based on previous research conducted at ISITV in Toulon France and NPS, and critical performance characteristics unique to this type of gliding underwater vehicle are examined. The mechanical and electrical systems required for vehicle operation are identified by subsystem, further decomposed to the component level, and their purpose and control requirements described in detail. As these systems are identified, development of the control architecture is accomplished by breaking down critical tasks and control requirements, while simultaneously evaluating the Sun SPOTs ability to execute those tasks.

E. THESIS ORGANIZATION

Chapter II reviews previous research conducted on this fundamentally different vehicle design, as well as research on more conventional gliding-vehicle technologies. Chapter III reviews previously conducted research and simulation development based on notional characteristics of the SeaDiver glider. The Autonomous Underwater Vehicle Workbench (AUVW) simulation software and the necessary interface requirements to integrate the physical model is also reviewed. Simulations and scenarios were previously

developed for the SeaDiver glider, and both might benefit greatly by determining correct response parameters from the physical model. Chapter IV describes several performance characteristics that must be taken into consideration when developing a vehicle prototype, a control system, and integrating each into the applicable simulation system. These concepts must be taken into consideration during every step of the development, testing, and analysis process. Chapter V outlines in detail the various individual systems and hardware characteristics of the vehicle, as well as discoveries made as candidate systems were being evaluated for implementation. Chapter VI completely describes the control-system development process and prototype control system operation. Chapter VII describes preliminary design and considerations for a test tank configuration, including interface considerations that have not yet been implemented. Chapter VII reports thesis conclusions and recommendations for future work.

II. BACKGROUND AND RELATED WORK

A. INTRODUCTION

The purpose of this chapter is to present the background for previous research conducted related to the SeaDiver glider, as well as to present an overview of related work in the field of gliding underwater vehicles. Projects that are currently employing the Sun SPOT as a control system are also examined.

B. BACKGROUND

This thesis began as a continuation of research conducted at the NPS by a team of visiting fifth-year French engineering students from University of Toulon ISITV(L'Institut des Sciences de l'Ingénieur de Toulon et du Var) in Toulon, France. The research entitled "Implementing A Low-Cost Long-Range Unmanned Underwater Vehicle: The SeaDiver Glider" by David Gassier, Jerome Rebollo, Romain Dumonteil, (Dumonteil, Gassier, & Rebollo, 2006) supervised by NPS's Dr. Don Brutzman, and ISITV Professor Didier Leandri, was completed in January of 2007 and unfortunately vehicle construction was never completed due to design shortfalls and budget limitations. There was also independent thesis research conducted at NPS simulating candidate missions for the glider, awaiting actual test data to validate projected performance characteristics. (Seguin, 2007)

C. NAVY UNMANNED UNDERSEA VEHICLE (UUV) MASTER PLAN

The Navy UUV Master Plan was originally published in April 2000 and an update was published in November of 2004 at the request of the Deputy Assistant Secretary of the Navy and OPNAV N77 (Submarine Warfare Division). The plan recommends missions, technologies, and programmatic recommendations for the future use of UUVs to support the Sea Power 21 initiative. Of the many conclusions and recommendations, the section of the plan most related to this thesis research is the call for commonality and modularity in UUV design. The plan recommends designing UUVs that share common core components such as control, navigation, sensor, and communication systems, as well as developing vehicles that support flexible configuration changes, such as adding new

payloads. The SeaDiver II Glider UUV investigated in this research presents itself as a potential solution to the need for vehicles supporting mission flexibility, and also evaluates a completely open-source control system option. (Naval Undersea Warfare Center, 2004)

D. SEADIVER GLIDER

There are a number of long-range gliding underwater vehicles designed and currently deployed. The uniqueness of the SeaDiver glider comes not only from its ability to conduct efficient long-range operations, but the scalability inherent in its design. When the payload requirements change for many of the current gliding vehicles, or something needs to be added, a current capability is lost, or the entire design must change and a new vehicle be constructed. Due to their design, they do not support scalable size or much mission flexibility. The SeaDiver glider, modeled after the airfoil shape, promotes rescaling the body size to accommodate various payload sizes without changing the remainder of the control and sensor system. This is possible when vehicle plus payload displacement remains neutrally buoyant. The most valuable property of the airfoil shape is that when scaled up or down consistently in all dimensions, its hydrodynamic behavior remains virtually unchanged. (Dumonteil, Gassier, & Rebollo, 2006)

Work from the technical paper was continued by the supervising Professor from ISITV in Toulon France, is referred to throughout this thesis as SeaDiver glider, and is shown in Figure 1.

Figure 1. SeaDiver Glider. (2007)

E. BLUEFIN ROBOTICS

Bluefin Robotics was formed from a core team of the Autonomous Underwater Vehicles Laboratory at the Massachusetts Institute of Technology. Bluefin is currently conducting research and development of a gliding underwater vehicle. The Bluefin Glider – SPRAY, is a buoyancy driven gliding underwater vehicle that utilizes a hydraulic pump to control its ascent and descent.

In water depths up to 1,500 meters, SPRAY is capable of traveling over 4,000 kilometers, over a period of approximately 6 months. The SPRAY gliding vehicle shown in Figure 2, is of the more common cyclindrical pressurized hull design.(BlueFin Robotics, 2008)

Figure 2. Bluefin Robotics-SPRAY Glider. (from BlueFin Robotics, 2008)

F. XRAY FLYING WING GLIDER

The XRay glider shown in Figure 3 is considered a high-performance undersea robotic vehicle that was developed in partnership between the Marine Physical Lab at Scripps Institution of Oceanography and The University of Washington Applied Physics Laboratory. The XRay glider employs a wing design that has a high lift-to-drag ratio. This wing design provides the ability to travel long distances efficiently as well as travel at higher speeds than existing gliders.

In July–September 2006, the XRay glider participated in its first at-sea test experiments in Monterey Bay. Vehicle specifications and test results have not been made available. (University of Washington, 2006)

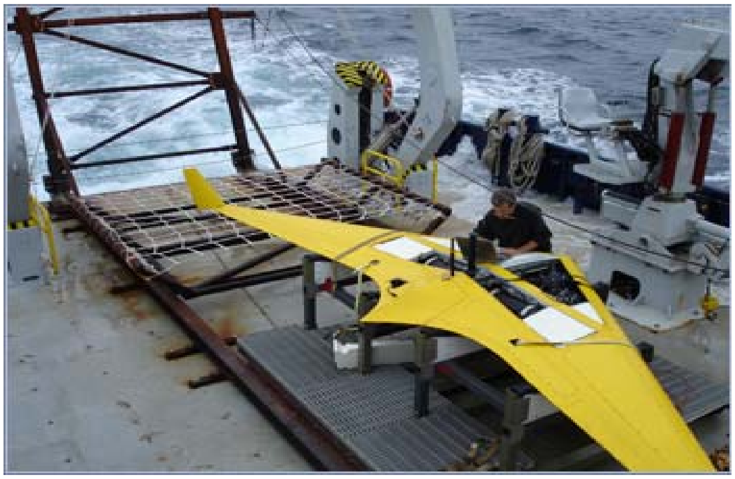


Figure 3. XRay Flying Wing glider. (from University of Washington, 2006)

G. RELATED PROJECTS UTILIZING SUN SPOT DEVICES

There are a number of projects utilizing the wireless sensor capabilities of Sun SPOT devices and employing the device as a robot controller. The two projects described below are current projects most closely related to this research.

1. Autonomous Light Air Vessels (ALAV)

The ALAV research emphasis is on autonomous and flocking behavior in a wirelessly-networked environment, yet is an early demonstration of the autonomous control capability of the Sun SPOT device. The ALAVs shown below operating in a warehouse are helium-filled vessels propelled by electric fan motors. Each vessel is controlled by a single Sun SPOT device to control the motors, communicate with other vessels, and monitor signal strength from the wireless radio to determine proximity to other vessels.

The vehicles are designed to behave in specific ways, yet are autonomous in their decision making process for locating and maneuvering towards other vessels. Figure 4 shows three ALAV baloons during operaton. (Berk & Mitter, 2006)

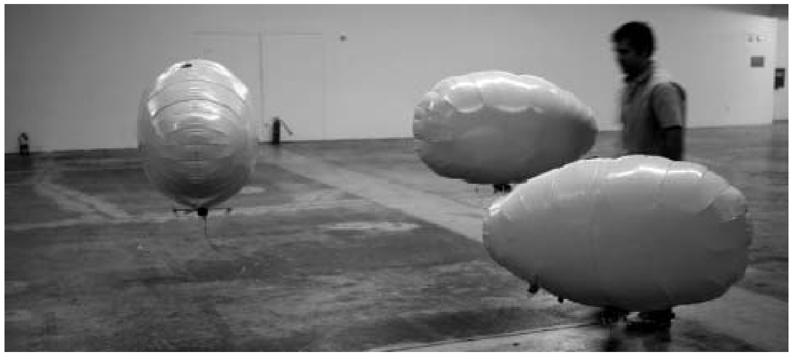


Figure 4. ALAVs operating autonomously. (from Berk & Mitter, 2006)

2. TrackBot

The TrackBot shown in Figure 5 is a robot platform developed by Systronix Corporation specifically for university-level education and research. The platform employs eight different sensor modules that provide object avoidance, beaconing, navigation and communication, and a single Java Sun SPOT device as a robot controller. This project is of specific interest as research is currently being conducted at NPS utilizing this platform in the modernization of a large-scale expeditionary warfare demonstrator. Also of importance is the demonstrated capability of the Sun SPOT device to read multiple sensors, make decisions based on those inputs, and actuate multiple motor controllers.

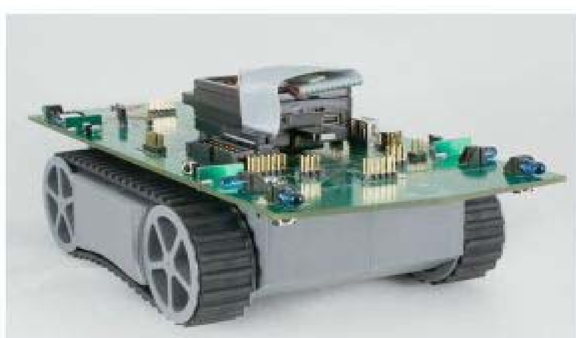


Figure 5. TrackBot robot platform. (from Systronics, 2008)

H. SOLAR POWERED AUTONOMOUS UNDERWATER VEHICLE (SAUV)

The Solar Powered Autonomous Underwater Vehicle (SAUV) project was developed by Falmouth Scientific Incorporated, in cooperation with the Autonomous Undersea Systems Institute (AUSI) and Technology Systems Incorporated (TSI) to meet the expanding requirements of AUVs to perform long-range missions and station keeping duties. The SAUV shown in Figure 6 is powered by an electric thrust motor. The SAUV operates at depths up to 500 meters and cruises at a speed of 1 knot with a maximum speed of 3 knots. The vehicle contains a battery system that provides approximately 1500 watt hours of stored energy. This battery system allows for deep water operation between charging cycles. The SAUV is normally programmed to perform deep operations at night, and charging operations during daylight hours. The solar panel provides 18 volt system power with a maximum power output of 85 watts.

This project is of particular interest to this research since the addition of a solar-panel to the top of the SeaDiver II glider might increase the mission duration and expand the possibilities for payload options with higher power requirements. A near-surfacce SAUV is shown in Figure 6. (Jalbert, Baker, Duchesney, Pietryka, & Dalton, 2003)

Figure 6. Solar-powered Autonomous Underwater Vehicle (SAUV). (from Jalbert, Baker, Duchesney, Pietryka, & Dalton, 2003)

I. SUMMARY

This chapter discussed the background of the previous research conducted related to the SeaDiver glider, as well as present an overview of related work in the field of gliding underwater vehicles, and projects that are currently employing the Sun SPOT as a control system.

III. SIMULATION ENVIRONMENT

A. INTRODUCTION

A major motivation of this thesis was to continue development of the SeaDiver gliding vehicle. Another motivation is to evaluate the Sun SPOT as a viable option to control the glider, and interface with existing simulation systems for testing and evaluation. This chapter describes the previous mission simulations that have been created, and the simulation software that needs to be developed for the control system interfaces with for mission rehearsal, replay, and any real-time control considerations.

B. SEGUIN SIMULATIONS

Thesis research was conducted to assess the effectiveness of glider vehicles at the Naval Postgraduate School and the results published in March of 2007 by John M. Seguin. Simulations were created using open-source simulations tools SIMKIT, VISKIT, and AUV Workbench all produced by NPS. The simulations examined candidate missions of tactical interest based on the notional SeaDiver glider. The results of Seguins' thesis research were promising but not conclusive enough to prove the ability of the SeaDiver glider to conduct tactical missions because many of the vehicle parameters and performance data were only estimates. (Seguin, 2007). Nevertheless the vehicle design, if validated, is expected to have significant tactical potential if deployable.

C. THE AUTONOMOUS UNDERWATER VEHICLE WORKBENCH (AUVW)

AUV Workbench is an open-source mission planning, rehearsal, and replay software package for autonomous vehicles, developed and managed at NPS, and can be downloaded along with all associated documentation from https://savage.nps.edu/AuvWorkbench. The software enables the user to create a mission profile based on a specific vehicle model and its actual physical characteristics.

Once the software has been downloaded and installed, a simple mission can be created based on the SeaDiver vehicle. There is a placeholder vehicle profile in AUVW for the SeaDiver glider. The properties for the vehicle were modified from the ARIES UUV model that already existed. (Dumonteil, Gassier, & Rebollo, 2006) Figure 7 shows the 2 dimensional (2D) X-Y plot of a basic dive-rise straight line mission.

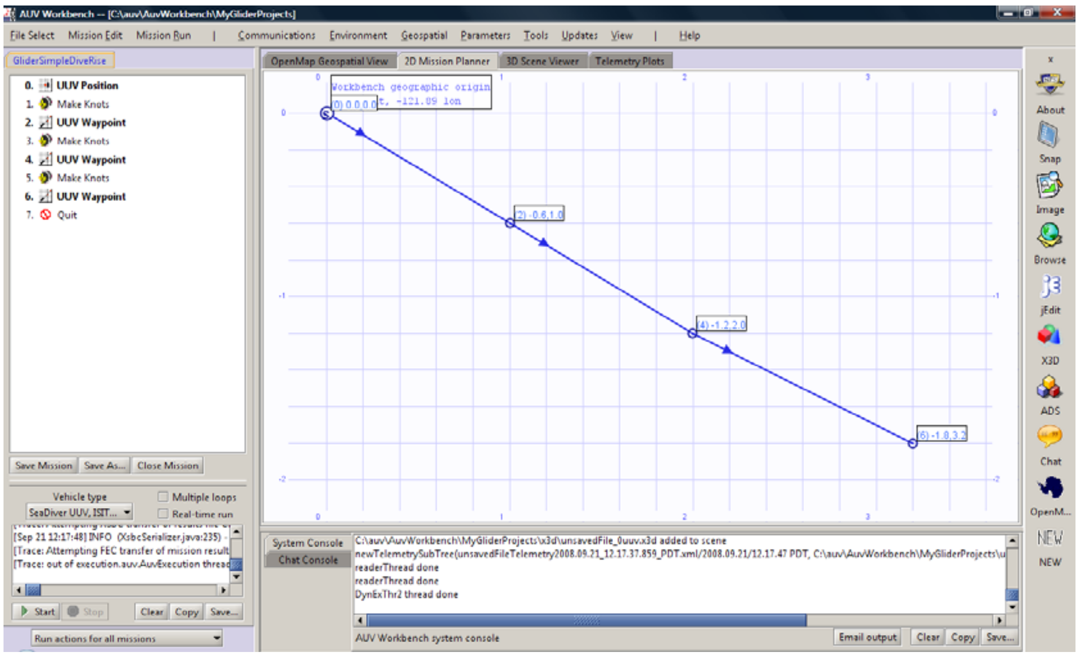


Figure 7. Screen shot of 2D mission planning for basic dive-rise test mission.

Once a mission is created, the mission profile can be exported in the Autonomous Vehicle Command Language (AVCL) file format. The AVCL file is an Extensible Markup Language (XML) based format that includes all aspects of the mission and vehicle information.

The AVCL file includes a section on mission preparation, and is followed by mission results. (Davis, 2006) The mission preparation portion of the basic mission described is shown in Figure 8.


```
  <?xml version="1.0" encoding="UTF-8" ?>
- <AVCL version="1.0" vehicleID="0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="AVCL.xsd">
  - <head>
      <meta name="description" content="AVCL output file generated by AUVWorkbench, execution-evel output file" />
    </head>
  - <body>
    - <MissionPreparation>
        <UnitsOfMeasure angle="degrees" distance="meters" mass="kilograms" time="seconds" />
        <GeoOrigin latitude="36.606998443603516" longitude="-121.88500213623047" />
      - <Configuration>
        - <UnmannedUnderwaterVehicle>
            <Capability value="thrusterPowered" />
          </UnmannedUnderwaterVehicle>
        </Configuration>
      - <UUVCommandScript>
        - <SetPosition>
            <XYPosition x="0.0" y="0.0" />
            <Depth value="10.0" />
          </SetPosition>
          <MakeKnots speedOverGround="false" value="1.2" />
        - <Waypoint>
            <XYPosition x="-0.6" y="1.0" />
            <Depth value="70.0" />
          </Waypoint>
          <MakeKnots speedOverGround="false" value="1.2" />
        - <Waypoint>
            <XYPosition x="-1.2" y="2.0" />
            <Depth value="10.0" />
          </Waypoint>
          <MakeKnots speedOverGround="false" value="1.2" />
        - <Waypoint>
            <XYPosition x="-1.8" y="3.2" />
            <Depth value="70.0" />
```


Figure 8. Example glider mission recorded as AVCL output file generated by AUVW.

A parser must be created to extract the critical mission information in a format best suited for import to the Sun SPOT. Since the controller is programmed in Java, parsing functionality might be programmed directly into the controller.

Actual mission data can also be imported in AVCL, or a number of other data formats for conversion, for 3D mission replay and analysis. The capability to generate a Hyper Text Markup Language (HTML) based mission report is currently being added. (Davis & Brutzman, 2005)

A major motivation of this thesis research was to find a suitable control system to support closing the loop between the virtual and real world. The initial intention was to create a basic mission in AUVW and develop a method for exporting the mission profile either in AVCL format for parsing in the control system, or integrate a parser into AUVW to facilitate exporting mission information directly to the Sun SPOT. It was discovered that neither of the communication protocols supported by the Sun SPOT

support file transfer to and from the Sun SPOT wirelessly. The only support there is for storing a data file in the Sun SPOT is to include it in the project folder when the controller software project is built and deployed to the Sun SPOT. There is however, API support for modifying an included data file once it has been deployed. It is recommended as the control system is refined, a mission update and modification interface gets developed to update mission profiles via wireless protocol. These missions are best described using AVCL.

This same process could be applied in reverse and employed once longer-range communication capability is added in order to export mission results and other data to the base station. Initially an interface could be developed in Java for testing, but ideally the replay functionality needs to be added to AUVW, and eventually get expanded to include real-time vehicle control.

D. SUMMARY

This chapter briefly described previous thesis research based on the notional vehicle that should be revisited once a suitable physical model in-the-loop environment has been established to refine vehicle parameters, and re-ran to establish vehicle mission capabilities. An overview of the AUV Workbench software suite, and recommendations for interface with a fully developed control system to establish a robust link between the virtual environment and the physical model was also provided.

IV. GLIDING VEHICLE DESIGN CONSIDERATIONS

A. INTRODUCTION

This chapter describes various design elements to be taken into consideration when developing a gliding underwater vehicle of this type. Much of the information was gathered during collaboration sessions with our French colleagues at ISITV based on findings during their initial period of research.

B. BEHAVIOR CONSIDERATIONS

Although it has been discovered that one unique feature of gliding type underwater vehicles is that many behavioral type rules are applicable to almost all glider types, the findings reported here are conservatively considered as applicable to the SeaDiver II. Energy in propeller-driven vehicles is inherent to the vehicle at the beginning of the mission, and can be employed for maneuver immediately. This is not the case in gliding vehicles. The energy in gliding type vehicles is not available until vertical lift has first been achieved, and then converted to forward motion by the wing shape. Until this has been achieved, there is no vehicle stability or control. Once forward motion has been achieved, that energy level must be managed constantly to capitalize on the efficiency needed for long-range operations. Energy management must be carefully analyzed as it relates to velocity.

With speed averaging perhaps 1.5 knots, gliding vehicles are considered to be "slow." This is not necessarily a limitation by design, as increasing the dive angle of any gliding vehicle will naturally increase the forward velocity. Due to the wing shape, and the dependence on it to convert lift to forward motion, this increase in velocity creates extra drag and therefore reduces vehicle efficiency. Careful planning of turns and other directional maneuvers is needed to maintain optimum efficiency. Each type of gliding underwater vehicle has relatively similar behavior, but the description that follows are findings specific to SeaDiver II and thus need to be carefully analyzed in control system development.

The dive-rise-turn cycle is shown in Figure 9 illustrates the typical vertical-sawtooth travel pattern. As the vehicle begins to dive, it is recommended the dive pitch

initially be set for approximately a 20 degree downward angle, then allow for the efficiency sensor system described in Chapter 5 to determine the most efficient angle to dive. If a turn is anticipated, the efficiency sensor system should be over ridden, and buoyancy reduced to increase the dive angle to gain forward momentum and build energy to power through the turn. For example, if the optimum speed is 1.2 knots, the angle should be temporarily increased to build speed to 1.5, or 1.6 knots. The amount the dive angle should be increased will need to be accurately determined as vehicle is tested. The dive-rise-turn cycle shown below in Figure 9 is compressed to illustrate the entire cycle. The actual dive and rise angles are much shallower.

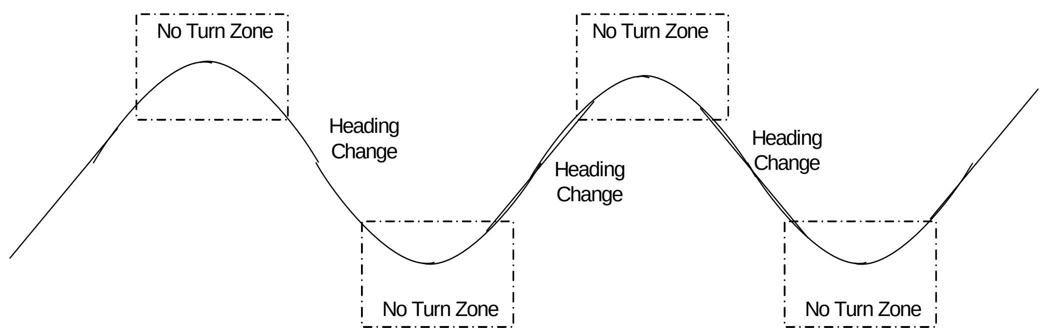


Figure 9. Dive-Rise-Turn Management Sequence.

Executing turns reduces efficiency and uses a much greater amount of energy than linear travel. It is recommended that heading changes are made in 20 to 30 degree increments per dive or rise cycle to avoid bleeding off too much energy and speed. This is also the case when the vehicle is transitioning from a dive to rise cycle, or vice versa. The boxes in Figure 9 that highlight the vehicle transition points should be considered "no turn" zones, and the vehicle should be level and stable through these zones, with the boundaries limited by the optimum operating speed. Once the vehicle has achieved this speed, a single increment of the turn should be executed, and timed with the dive cycle to ensure the vehicle returns to a stable state at the optimum speed to execute the transition back to the rise cycle. Figure 9 depicts the requisite number of dive and rise iterations for executing a ninety degree turn. (Leandri, 2008)

C. VEHICLE BODY

The body design of this type glider is a free flood design. The inherent advantages to this type of vehicle design is that varying the payload size does not affect vehicle design parameters, and when building the body, the challenge of trying to maintain watertight integrity is not an issue. Due to the shape of the vehicle, there are a substantial number of sharply angled seams that would normally be difficult to maintain a seal at any pressure. The free flood design also eliminates any need to take hull pressure into design consideration, therefore enabling the builder to use a number of inexpensive materials for body construction. The SeaDiver II is constructed mainly of plywood and fiberglass, and since the pressure is equalized both inside and outside the hull, it remains effective at any depth.

Another design consideration that was discovered via trial and error was the overall length of the body. When the vehicle length is anything less than 1.5 Meters, the vehicle body is extremely unstable.

Ideally the vehicle body should be no less than 2 meters in length, and the remaining dimensions scaled to match. There has been no evidence to suggest that there is a maximum possible vehicle length. The dimensions of the successfully tested ISITV version of the SeaDiver are shown in Figure 10.

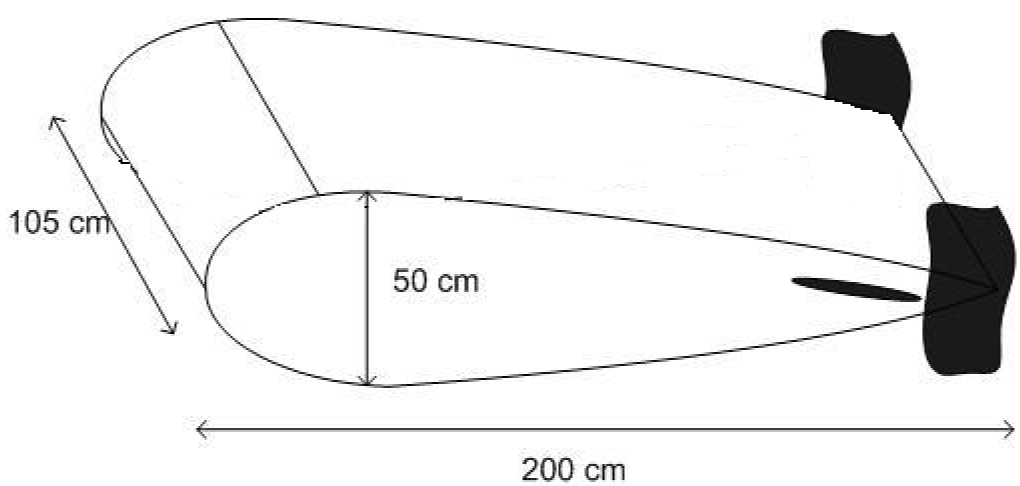


Figure 10. SeaDiver II overall dimensions.

D. BOUYANCY

When developing the buoyancy system for this underwater vehicle, the first determination that must be made is the mission of the vehicle, and maximum required depth of the operating environment. It was discovered by the ISITV team that for depths greater than 100 meters, any ballast systems should be hydraulic and oil based, vice pneumatic and air-based, since the latter tends to act less accurately at greater depths. It is recommended for simplicity of design that the initial prototype vehicle employ a pneumatic air-based ballast system. Another factor that must be taken into account is the operating pressure of any pumps. The two piston type pumps described in Chapter 5 to be employed as a trim ballast system are limited to approximately 60 meters. It is also recommended that once a successful tank test of a ballast system has been achieved, all rubber or plastic tubing be replaced by flexible copper tubing and fittings to eliminate any possible malfunction due to increased pressure collapsing the transfer lines.

E. SUMMARY

This chapter provided an overview of three functional areas in which special consideration must be taken when developing a gliding vehicle of this type. Vehicle behavior and energy considerations were discussed, as well as special considerations to be taken when constructing the vehicle body, and then developing a suitable buoyancy system.

V. VEHICLE CONSTRUCTION AND DEVELOPMENT

A. INTRODUCTION

This chapter describes the origination of the SeaDiver gliding vehicle design, construction of hardware components, and the evolution of the vehicle throughout the research period. It begins with findings made when exploring buoyancy solutions and work conducted while attempting to construct the original vehicle. Also described in detail are the various systems that were employed, or were investigated as potential options for employment, based on the model turned over to NPS by the team at ISITV.

B. VEHICLE DESIGN

The uniqueness of the SeaDiver Glider resides in its physical design. As shown in Related Work Chapter II, there are numerous vehicles employing the buoyancy-driven gliding design to achieve efficient long-range operations. These vehicles are expensive to build, and require major redesign in the event the intended mission or payload requires a major change. The SeaDiver glider body is constructed from low-cost materials. As the vehicle size requirements increase, only the body must be reconstructed as long as payload buoyancy remains relatively neutral. This feature results in flexible and economic modifications despite major mission changes.

1. Original Design

The original SeaDiver Glider design, as described in (Dumonteil, Gassier, & Rebollo, 2006) never actually came to fruition. Many design considerations were described in detail, some with different variations. Due to budget and time constraints, system-level testing and final vehicle construction were not completed.

2. Status at Project Takeover

The original SeaDiver parts and framework were recovered from the NPS AUV lab. There was a primitive framework, and numerous parts, but the Glider was nowhere near assembled or operational. When taking over the original attempt at the vehicle, a lab station was established, and the focus of research became trying to assemble the original vehicle based on the technical document. There were no systems intact as described in

the paper, and many of the concepts had not been proven. This slowed progress significantly, as the author had no previous experience in underwater vehicle design.

The design of the vehicle body is described in great detail in the original technical report and is illustrated below in Figure 11. The body dimensions were based off of a predetermined airfoil design ratio. The vehicle body is based on an airfoil that is symmetrical, has no camber, and has a 22% thickness to chord length ratio, meaning it is 22% as thick as it is long. (Dumonteil, Gassier, & Rebollo, 2006)

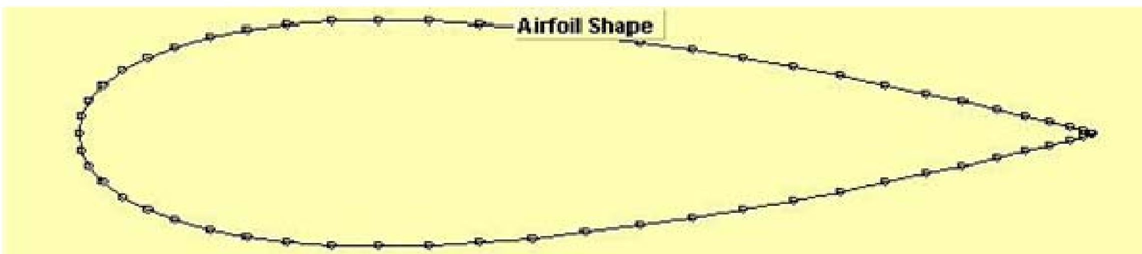


Figure 11. Airfoil Shape of the SeaDiver Glider. (from Dumonteil, Gassier, & Rebollo, 2006)

The initial attempt at the vehicle body is shown in Figure 12. The dimensions were in accordance with the original design as:

Length: 1.60m (63")

Width: 0.80m (31.5")

Figure 12. Gassier vehicle body.

The first design challenge was to identify a suitable buoyancy solution for the vehicle. There were multiple possible solutions outlined in the technical report, but no conclusive system level testing had been performed or documented.

Rudimentary testing began using a potable water compression tank that was included in the parts inventory recovered from the lab.

The compression tank is normally employed to prevent potable and irrigation well pumps from losing suction pressure. This is accomplished by forcing an internal bladder full of water as it compresses the air in the tank as shown in Figure 13.

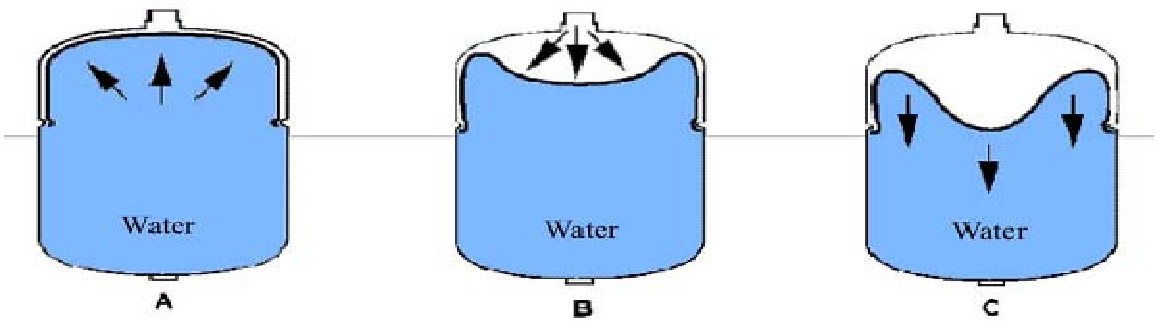


Figure 13. Expansion tank increasing buoyancy by displacing water with compressed air. (from Dumonteil, Gassier, & Rebollo, 2006)

The initial description recommended using a reversible air pump to cycle air from a compression tank, forcing the water out of the tank to allow the vehicle to rise, and thus cycling the air from the expansion tank back to a separate compression tank to draw in water and complete the dive cycle. This approach is illustrated in Figure 14.

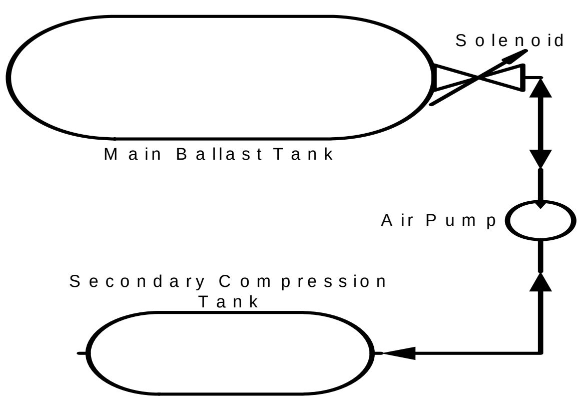


Figure 14. Pneumatic air-based buoyancy system.

Our first attempt at developing a suitable buoyancy system was based on a variation of this theory. Instead of using an air pump to cycle air between two tanks, our theory was that the air portion of the tank might be pressurized, and a submersible water pump might then pump the tank full of water to submerge, with a solenoid valve maintaining the water in the ballast tank. Similarly for the rise command, the solenoid valve might simply open and allow the water to escape as shown in Figure 15. As the air pressure in the tank increased, it forces the water back through the pump, thus reducing pump motor cycles from one to two, and greatly increasing electrical efficiency.

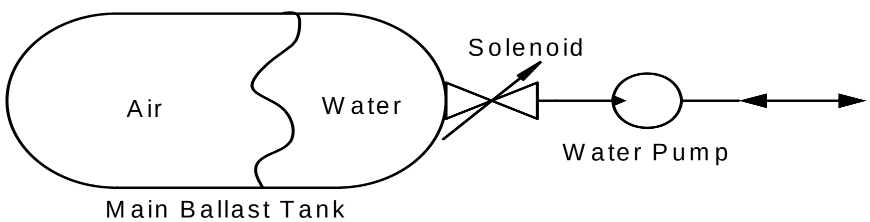


Figure 15. Water pump-based experimental buoyancy system.

The first test of this theory was conducted by obtaining a basic submersible pond pump from a hardware store and connecting it directly to the expansion tank as shown below in Figure 16.

Figure 16. Submersible pump attached to expansion tank.

A standard yard waste container was filled with water, and the pump connected to outlet power. This might not have been a suitable pump to employ in the glider, but was an economical proof of concept. The initial expansion tank pressure was set to 5 pounds per square inch (psi) to allow the pump to quickly overcome the pressure and fill the tank with water, causing submersion. Power was applied to the pump, and the tank gradually fell to the bottom of the test tank. Due to a lack of solenoid valve to maintain the water in the expansion tank, when pump power was removed, the air was allowed to expand and forced the water from the tank, and the ballast tank returned to the surface.

The next step was to begin researching technical specifications to determine if this method might be able to perform at the desired depths, and also learn what air pressure would be required in the tank to be able to force the water back out. This then creates a requirement for a suitable pump capable of overcoming that air pressure to fill the tank and allow the vehicle to submerge. Not having the suitable background, an initial consultation was made with a hydraulic engineer to discuss possible options. As discussed below, research and investigation into the buoyancy system, and this vehicle design was suspended after reviewing findings made by our colleagues in France.

3. ISITV Glider

In trying to establish a more concrete starting point, the second supervisor of the original design work was contacted at the University in Toulon France. Professor Didier Leandri had also been continuing research on the SeaDiver glider design, and had made substantial findings and progress.

Upon hearing that research on the SeaDiver glider was starting again at NPS, Professor Leandri extended an invitation to visit his lab at ISITV in Toulon to share findings and foster the research relationship between ISITV and NPS. When visiting the lab at ISITV, NPS provided demonstrations of the updated AUV Workbench software, and the newly available Java Sun SPOT. The remainder of the visit was spent taking measurements, and sharing research findings with lab personnel.

Professor Leandri and his team had made substantial progress and conducted extensive testing on the original design, making a number of discoveries regarding the original design that had failed during test and evaluation. The most substantial of these

findings was the instability found in the original vehicle length. The original design called for a vehicle dimension of 1.6 meters. It was found that at this length, the glider performed erratically and was impossible to stabilize. The length dimension was increased to 2 meters, and this changed provided the needed stability to maneuver. As the vehicle design is based on the scalable airfoil design, this modification also changed the other dimensions proportionally.

The ISITV research team also found that although they had a “suitable” control system in a 386 Mhz Processor, programmed in machine language, to capitalize on the versatility and functionality of the glider design, a more robust system was required. Machine language is not suitable for a flexible mission-oriented robot controller.

The vehicle speed through water was found to be 1.2 knots, and this value is used for the remainder of our design considerations.

By the end of the four day visit, Professor Leandri had determined that our ambition to continue progress in this important area of research warranted providing the existing vehicle his team had been working on to NPS to continue. Documentation began on the systems and parts that were available to continue progress while awaiting shipment of the vehicle to NPS.

The hull is shown in Figure 17 and the dimensions of the glider body are as follows:

Length: 2 meters
Width: 1.05 meters
Height: 50 centimeters
Rear fins: 58 cm (H) 21cm (W) ~ 2.5cm thick

Body construction is primarily out of plywood, and at the time of measurement, the control fins were not installed, but are estimated to be 10 inches long and resemble surfboard fins.

Figure 17. SeaDiver II hull.

C. VEHICLE CONTROLLER

Assessment of the original SeaDiver design by NPS and the ISITV lab found that this type of vehicle lacked an electrically and economically efficient control system to support long-range operations, especially if the vehicle was to be considered for long-term tactical employment. Only summary technical specifications of the original proposed controller were available, and we were unable to obtain any of the original code for the control system. We were also unable to obtain the original test and operational code from the team at ISITV, since the controller had been removed and repurposed to continue testing on another project. This loss made the design of a new suitable control system more challenging since there was not a system to duplicate.

From the limited information provided, initial research was unable at first to find a suitable solution for a control system. The ISITV team employed a 386 Mhz Processor based controller that lacked the needed robustness, and was found to be cumbersome to update software and mission profiles. The replacement solution evaluated by this research is the employment of the Sun Microsystems Java-based Sun SPOT (Small Programmable Object Technology).

1. Sun SPOT

Sun Microsystems started Project Sun SPOT in late 2004 to perform wireless sensor technology research and development, while also developing a hardware platform to test a small flexible Java Virtual Machine (JVM) named Squawk. The Sun SPOT devices shown in Figure 18 were a product of this research, and were released to the public late in 2006. The Sun SPOT is a small-footprint wireless-capable sensor platform.

Figure 18. Free range Sun SPOT and base station.

The Sun SPOT main processor is an Atmel AT91RM9200 system on a chip (SOC) integrated circuit. The unit incorporates the ARM920T ARM Thumb processor, based on the v4T ARM architecture. The main processor resides on the main board, but the majority of the functionality of the Sun SPOT resides in the connected daughter board. The Sun SPOT is shipped with the eDemo board, and although there are schematics and API support for a number of additional function-specific boards, none are currently being produced. The eDemo board contains an Atmega88 processor, a small amount of flash memory, light and temperature sensors, a 3-Axis accelerometer, eight tri-color light emitting diodes (LEDs), and two momentary pushbutton switches.

Of primary interest to this research is the 3-axis accelerometer and the general purpose input output (GPIO) pins. All are accessible from provided libraries included in the Sun SPOT software development kit (SDK).

The GPIO pins are located on the top of the Sun SPOT via a 20 pin connector. The pinout for the connector is as shown in Figure 19.

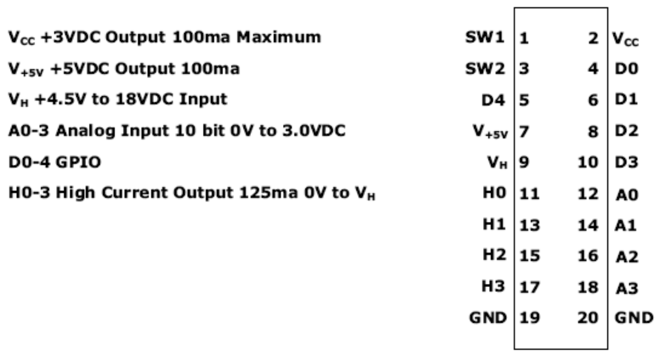


Figure 19. Sun SPOT top connector. (from Sun Microsystems Inc., 2007)

As depicted in Figure 19, there are four GPIO pins that can be configured as either input or output pins. These pins have a three volt maximum output and utilize the Universal Asynchronous Receive Transmit (UART) protocol which governs serial communication. Pins D0 and D1 can be configured via software as UART receive and transmit lines respectively, and D2 and D3 can be configured as I2C-DATA and I2C Clock respectively. H0 through H3 are considered high output pins, and the high output voltage is set by a voltage between 4.5 and 18 volts being connected to the Sun SPOT via pin 9 (Vh). Pins A0 through A3 utilize the analog to digital conversion capabilities of the processor and accept analog signals from 0 to 10 Volts.

As previously noted, the eDemo board also contains a 3-axis accelerometer. The accelerometer is an ST Microsystems 3-Axis 2G/6G Inertial Sensor. The accelerometer outputs three voltages that are converted and available to the programmer via the software API.

D. BUOYANCY CONTROL

Besides providing the basic rise and dive functionality, buoyancy can also used to control the roll angle of the vehicle attitude. The current design accomplishes this via two separate systems, but further refinement may reveal a method to combine the two for increased efficiency.

1. Main Ballast

The main ballast system consists of a pump and associated plumbing to transfer air or oil between a compression bladder and an expansion bladder. The two system

components are depicted in Figure 20. In the ISITV vehicle, the interconnecting tubing was common garden hose, but the recommendation was made by ISITV that this tubing was not really suitable for great depths since the resulting water pressure in the free-flood hull would tend to compress the tubing and restrict flow. Instead, these hoses need to be replaced by copper tubing or steel jacketed hoses.

Figure 20. Expansion bladder for main ballast and associated tubing.

Figure 20 shows the expansion bladder that is placed in the forward end of the vehicle. The tubing connects the expansion bladder to the main ballast pump, and then to the compression bladder shown in Figure 21.

Figure 21. Compression bladder for main ballast.

As air or oil is pumped from the expansion bladder at the forward end of the vehicle, the bladder deflates making the vehicle less buoyant, and the dive process begins. Likewise, when the process is reversed, and the air or oil is pumped from the

smaller compression bladder to the expansion bladder, the vehicle nose becomes more buoyant resulting in the rise portion of the cycle.

The pump that was employed in the ISITV vehicle is shown below. Actual technical documentation for the pump was unable to be located, and other than the 12 volt input required, no further information was able to be retrieved from the nameplate shown in Figure 22.

Figure 22. Main Ballast Pump.

The solenoids used in the main ballast system in the ISTV vehicle are manufactured by Danfoss Corporation in Denmark and are shown in Figure 23.

As with the ballast pump, the printed technical documentation was unable to be located, so the following specifications are provided from the information plate on the device:

- Part #: 018F6856
- Voltage: 12 Volts DC
- Power: 20 Watts

Figure 23. Ballast System Solenoids.

2. Trim Ballast

Vehicle trim is accomplished using two piston-type positive-displacement water pumps made by Robbe Modellsport Corporation shown in Figure 24. The trim ballast pumps are placed in the vehicle on both the port and starboard side and can be controlled independently. Alternatively, to reduce the number of controller pins required, each might be operated in tandem by inverting the control line signal from one to control the other.

One precaution to take into consideration when utilizing this option for trim ballast is that these pumps are only rated to 60 meters in depth. If the vehicle mission profile is to exceed this depth, the current equipment for this air-based ballast system will not be suitable.

Figure 24. Trim Ballast Piston Tank

E. NAVIGATION

Glider navigation between waypoints is accomplished via a system comprised of separate devices to provide the initial surfaced location, heading and depth. Once an accurate global positioning system (GPS) fix is no longer available due to submersion, the control system must rely on the remaining sensors to gather information for dead-reckoning estimation of position until returning to the surface and obtaining another fix.

1. Location

When the vehicle is being initiated, and is in suitable range of the surface, it is desirable to acquire an accurate GPS fix to determine vehicle position, and correct for navigational errors accumulated in the dead-reckoning process. When investigating suitable GPS units, a unit was needed that had low power consumption and a separate low-profile antenna module for mounting on top of the vehicle without impacting hydrodynamic performance. The final requirement was to find a GPS receiver able to communicate via universal asynchronous receiver/transmitter (UART), connected via designated Sun SPOT input/output pins D1 and D0.

The device selected for testing and evaluation that met all requirements is the ET-202 GPS Receiver Engine Board produced by USGlobalSat™ and purchased through SparkFun Electronics. Multiple views are shown in Figure 25.

Figure 25. ET-202 GPS Receiver Engine Board. (from SparkFun)

The GPS Engine board is based on SiRF© GPS architecture and utilizes their StarII high-performance and low-power consumption chip set. Specifications for this GPS receiver are shown below in Figure 26.

Accuracy:	Position Horizontal 15m
Time:	1microsecond synchronized to GPS time
Datum:	WGS-84
Voltage supply:	3.3Vdc ~ 5.5Vdc
Current supply:	Continuous mode 60mA Trickle power 25mA typical
Backup Power:	+2.5V to +3.6V
Backup Current:	10uA typical
Serial Ports:	One for GPS, one for DGPS
Electrical level:	TTL level, Output voltage : 0 ~ 2.85v
Communication:	Full duplex asynchronous
Code type:	ASCII
GPS Protocol:	SiRF binary/NMEA 0183 changeable(Default:NMEA)

Figure 26. GPS device characteristics. (From USGlobalSat)

The system utilizes a remotely placed 3 volt active antenna that requires a supplied adapter to connect to the MicroMate Connector (MMCX) on the GPS receiver board.

The antenna acquired was from SparkFun electronics and is a 3 Volts Magnetic Mount Active GPS Antenna Manufactured by ONSHINE and is shown below in Figure 27.

Figure 27. Surface Mount GPS Antenna. (from SparkFun)

In accordance with the provided Sun SPOT technical documentation, solder connections were made between the GPS engine board serial UART pins and the UART pins on the top connector of the Sun SPOT. By default the GPS engine transmits the NMEA sentence data at 4800 bits per second, so the Sun SPOT test code was set to

initialize the UART to the same data rate. A simple test function was written to read data from the UART receive pins on the Sun SPOT and output the information to the system console for evaluation.

The antenna was connected, associated power provided as shown in Figure 28, and placed in an open location with unobstructed access to sattelite signals. (USGlobalSat)

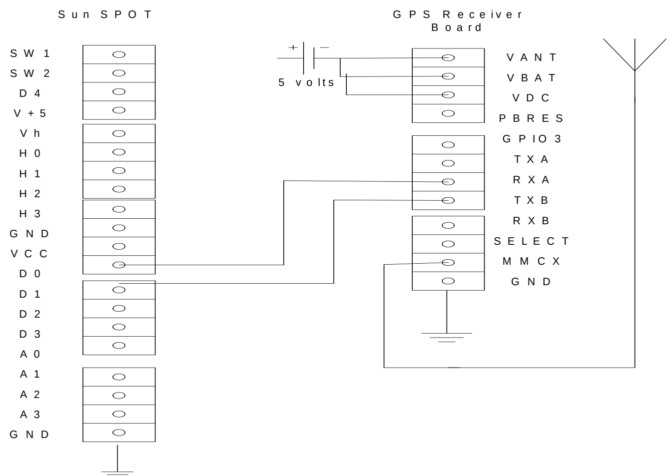


Figure 28. Sun SPOT and GPS Receiver Board connection diagram.

The first attempt at reading data from the GPS engine board was unsuccessful so troubleshooting began to determine whether the problem was due to the GPS board, or the Sun SPOT.

A message was posted on the Sun SPOT developers technical forum at https://www.sunspotworld.com/ forums that included the test function. The reply from Sun indicated that the code appeared to be correct, and further recommended a loop back connection be made between the Sun SPOT transmit and receive pins to ensure it could read data from the external interface. The test code was modifed to transmit ASCII characters and then receive and print them to the system screen. The loop-back test was successful. Test code is saved in AUVW version control, and listed in Appendix E. Further investigation into postings made to the technical forums revealed that a large number of developers had been encountering the same difficulties communicating via

UART with the Sun SPOT. A significant amount of research time was devoted to the troubleshooting process. Sun followed up with an admission that as the UART API was a recent capability addition to the library that there were still a number of issues that affect communication at standard data rates, adding that corrections were being made to fix the Sun SPOT UART code. This code became available in September 2008 in the developer release of the Blue SDK Version, as this thesis reached completion. Initial tests using the revised Blue distribution proved successful.

2. Heading

Accurate vehicle heading information is critical to vehicle navigation. Although the internal gyroscope on the Sun SPOT may be used to detect changes in heading utilizing that data in the dead-reckoning process, initial heading information must first be obtained. The ISITV glider utilizes three 3-axis gyroscope to provide inertial navigation information. While the investigation into possible options continues, for testing and evaluation purposes the heading value in the Sun SPOT controller is set to zero, and the Z-Axis accelerometer output is used to track heading changes.

3. Depth, Pitch, and Roll

Depth information is gathered by three piezoresistive transmitters placed on the underside of the vehicle. Two sensors are placed port and starboard on the forward end, and one at the center of the aft end. Three sensors not only provide redundancy, but can also be used to collect and confirm vehicle attitude information to be compared to gyroscope data for error checking. The differential in the forward two sensors can provide vehicle roll information, while the differential from forward to aft can be used to estimate vehicle pitch. Test results by the French team confirmed the viability of this approach.

The conversion of the difference in depth from each sensor can be derived using simple trigonometry for right triangles. As shown in Figure 29, the distance between the aft depth sensor, and the corresponding midpoint between the two forward sensors, is substituted into the equation for the sin of a triangle as the hypotenuse.

The difference in depth value between the forward and aft sensors is similarly applied as the side opposite angle theta. The result is the vehicle pitch. The same derivation is easily applied to the port and starboard sensors to compute roll information.

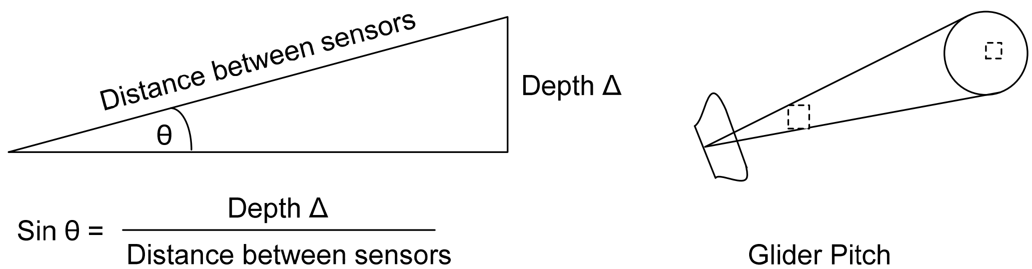


Figure 29. Pitch determination utilizing depth-sensor readings, from midpoint between forward sensors and aft sensor.

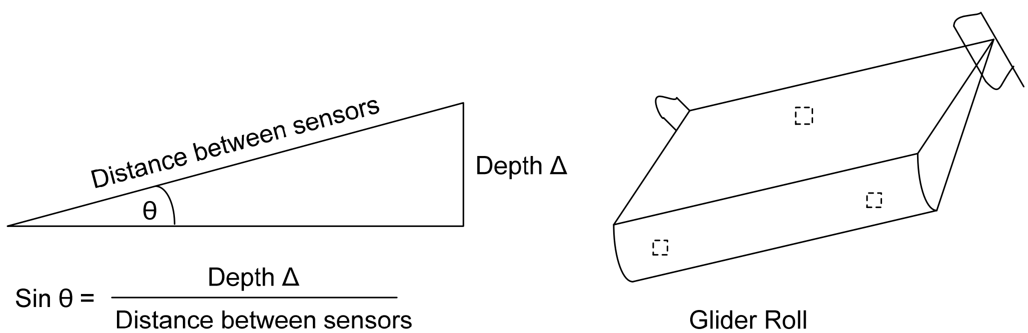


Figure 30. Roll determination utilizing depth-sensor readings, measured between port and starboard forward sensors (shown using dotted lines).

One of the transmitters is shown in Figure 31. It is manufactured by Keller-Druck, a European manufacturer of precision depth sensors and transmitters. Specifications are as follows:

Type: PA-23S/80565.55

Range: 0-30 bar

Output: 0-5 Volts linear

Vcc: 8-28 Volts DC

Figure 31. Keller-Druck Piezoresistive Depth Transmitter.

The initial challenge in using these sensors (and a number of other sensors) is that the output is a 0 to 5 volt range, and the input to the Sun SPOT GPIO pins is limited to 3.2 volts. A possible solution to overcome this difference is to create a linear resistive voltage divider network between the sensor output and the Sun SPOT input pin. As the sensors have not arrived, a substitute voltage is created and adjusted by utilizing a simple variable potentiometer to simulate depth changes by varying the input voltage to the Sun SPOT analog to digital GPIO pin A1. The connections are shown in Figure 42.

F. DIRECTIONAL CONTROL

Although the trim ballast might be utilized to assist in turning, heading changes and corrections are made utilizing two servo-driven control planes located on the aft portion of the vehicle on the port and starboard sides. The control planes might be configured to utilize a single control signal from the Sun SPOT with that signal inverted so that the plane movements oppose each other. Since independent plane control is likely desired to also assist in pitch changes, two control signals will likely be needed.

G. DATA STORAGE

The Sun SPOT has a built in flash memory with a 4 megabyte (MB) capacity. This memory is also used to store deployed applications and program data, therefore making it inadequate for the storage of any sort of data collected, or extensive mission profile information, thus an external data storage solution is also necessary. There are two potential options, one of which was investigated in this research. The second is newly emerging and definitely warrants further investigation

1. Universal Data Logging Device

The method that was investigated in this research was to utilize a small footprint Secure Digital (SD) data storage device used in many electronic cameras. These devices are inexpensive and store large amounts of data (available up to 1 giga-byte). The device selected was the Logomatic V1.0 Universal Data Logging Device produced by SparkFun Electronics Corporation and is shown in Figure 32.

It was chosen due to the simplicity of operation, low cost ($60), and simple Universal Asynchronous Receiver Transmitter interface (UART).

Figure 32. Universal Data Logger holding SD memory card. From (Sparkfun)

In accordance with the technical documentation the SD card was formatted to the FAT-16 file format on a personal computer prior to initialization of the device. Solder connections were made between the transmit and receive pins of the data logger to the corresponding D0 and D1 pins of the Sun SPOT. Ground and power connections were made in accordance with the specifications.

The same loopback test code that was utilized to troubleshoot the connection to the GPS engine board was modified to simply write data to the logging device, and deployed to the Sun SPOT, this code was later reduced to the same loopback test code listed in Appendix E. Note that the latest release Sun SPOT SDK Blue version has completely changed the UART interface and this code may no longer function. The logging device was initialized and all visual indications were normal. After a period of two minutes, the logging device was stopped, and the SD card inserted into the laptop for inspection of the log files. As expected, the device had created the appropriate log file, but there was no data present. Inspection of the process utilizing a logic probe indicated data bits were being transmitted from the Sun SPOT. The data rate was changed on both

the Sun SPOT and the logging device, and after multiple attempts, and a significant amount of time troubleshooting, logging data via UART on the Sun SPOT remained unsuccessful. It appears that the inability to log data at this point may be attributed to the same deficiencies in the UART interface as encountered when trying to communicate with the GPS engine board. The Sun SPOT libraries provide an API for reading and writing to SD memory, but it was written to specifically support the eFlash board.

2. eFlash

An external independently produced alternative to the SD data logger described above is now emerging. Sun Microsystems developed a number of experimental daughter boards with more specific functionality that replace the eDemo board on the Sun SPOT. One of these boards is an SD flash drive. A major advantage to utilizing this type of device is that the software interface provided in the Sun SPOT SDK was written specifically to support this device, hopefully eliminating the existing difficulties communicating with devices serially via UART. Although this requires its own dedicated remote Sun SPOT for operation, the radio communication and unused processing resources might also be capitalized on to assist in other control functions. Initial review of the API and documentation also support this approach as a better option than the Universal Data Logger, since it supports any data type desired, as where the Data Logger is limited to ASCII text.

Another potential advantage to utilizing this device for data storage is that Sun Microsystems recently released the hardware schematics and device software open source, making utilization of these devices a more economical and robust option. (Sun Microsystems, 2008) Sun Microsystems elected not to produce these devices, and no other manufacturer is currently producing them. Time and resource constraints inhibited our ability to produce a complete configuration for further testing and evaluation.

H. COMMUNICATION

Another inherent feature of the Sun SPOT that presents a robust solution to existing shortfalls in UUV development is the built in wireless communication functionality. A problem that currently plagues vehicle testing and operation is the need

to remove the vehicle from the wet environment and open it up to make even minor changes to the control software, or vehicle mission profile. Waterproof and pressure-tight connectors exist for communicating with the controller through the vehicle body, but evaluation of those connectors by the ISITV team revealed that they are often expensive, and can degrade rapidly in an ocean environment. These connectors also mandate the addition of leak-detection systems, adding further overhead to an already complex engineering task.

The Sun SPOT wireless network communication system uses an integrated radio transceiver, is IEEE 802.15.4 compliant, and operates in the 2.4GHz to 2.4835GHz range. This enables communication to host applications running on a personal computer via the wireless base station, or between remote Sun SPOTS. There are two different communication protocols provided in the API that support either radio datagram, or connection oriented communication. Sun SPOTS also possess the capability to act as wireless hop relay points as a routing protocol is also provided. This communicationfeature supports code deployment, data collection and dissemination, and provides several robust solutions to current communication limitations. (Sun Microsystems Inc., 2007).

In the early stages of vehicle development and testing, the wireless communication feature can be exploited to make code changes, collect performance data, and inject virtual sensor data without removing the vehicle from the test tank and apparatus. In an open-ocean operating environment, wireless links might be used to communicate with the vehicle when surfaced to collect data and update mission-profile instructions. Conceivably the wireless protocol might also be exploited for inter-vehicle communication supporting multiple vehicle operations, but surface rendezvous is a prerequisite step.

The wireless communication antenna is an inverted-F antenna printed on the top layer of the Sun SPOT main board. Initial testing and evaluation in an unobstructed environment revealed the maximum line of sight range for reliable communication was approximately 30 meters. This is not a problem within the vehicle for inter-controller communication, but presents a number of challenges for external communication. Since the controllers are housed in a metal dry box, an external antenna is a necessity. Along

with the need for the external antenna, an amplification solution will also be required to increase the vehicle communication range to support operations at sea. The Sun SPOT documentation states that certification does not allow an external antenna to be connected to the Sun SPOT.

I. POWER

When developing a gliding underwater vehicle to support long range autonomous operations, power management and monitoring are paramount. The Sun SPOT is an electrically efficient controller option, that minimally taxes the main electrical supply. For testing and evaluation purposes, the Sun SPOT internal power management was evaluated as a separate system from main vehicle power.

1. Sun SPOT Power

The power circuit onboard the Sun SPOT charges the battery, regulates the power to the main board, eDemo sensor board, and regulates power during deep-sleep operations. The power controller also measures the battery voltage, the battery charge current and discharge current, Vcc voltage, Vcore voltage, Vext external voltage, and USB voltage Vusb. The Sun SPOT continuously monitors these voltages and indicates any deviation from the acceptable ranges on the power status light emitting diode on the front of the device. These voltage levels and status are also available to the controller program via the API. This is not only a valuable attribute for system monitoring during normal operation, but can also provide invaluable testing and evaluation data to determine vehicle duration before conducting open-ocean missions. As the entire system has not been constructed, and all devices connected that might affect Sun SPOT power consumption are not yet connected, accurate efficiency and duration data could not be obtained. (Sun Microsystems Inc., 2007)

2. System Power and Monitoring Requirements

The vehicle power is supplied by 12 Volt batteries that supply power to the controller as well as all sensors and actuators. No power budget has been estimated.

Monitoring the power status not only provides important information about vehicle duration during testing, but remains a critical task during normal operation, since a power failure might prevent the vehicle from surfacing and result in a total loss. The current design utilizes a power controller and monitoring device that monitors battery voltage and current. This information must be provided to the controller for monitoring and logging. The design specifications of such a device are unknown at this time, but it remains advisable to investigate into the possibility of setting a threshold in the system power controller redundant to that in the Sun SPOT to trigger a separate emergency-surface system in the event of power system failure.

As vehicle refinement progresses, consideration needs to be given to adding solar charging capability in the form of a flexible solar mat affixed to the top of the vehicle. A suitable product would have to be found that can withstand the saltwater environment, and testing needs to be performed to determine if enough light could be collected from a hover position and depth to keep the vehicle undetectable from the surface. Broached operation may also work satisfactorily as evidenced by the SAUV described in Chapter II. (Jalbert, Baker, Duchesney, Pietryka, & Dalton, 2003)

J. EMERGENCY SURFACE

1. System Description

The vehicle design that tested with positive results by the ISITV team included an uninflated personal flotation device in the body of the vehicle. When the processor experiences any number of emergency conditions, or a lack of sufficient power remaining to re-inflate the ballast, a solenoid might then actuate inflating the personnel flotation device, increasing the buoyancy and forcing the vehicle to the surface. Another option to consider could be to develop a relay attached to the power management system that activates the emergency surface system if the monitored battery power level drops below a predetermined threshold. Desert Star Systems also provides a low-cost solution that can be triggered by an external sonar transponder. (Desert Star Systems, 2008)

K. CONTROL SYSTEM CONNECTION

The number of sensors and actuators that are needed to control the vehicle requires that the control system be comprised of at least two Sun SPOTs. Communication between the two Sun SPOTs is accomplished utilizing the wireless functionality due to the limited number of UART capable GPIO pins (two per Sun SPOT).

Since the GPS and an external memory solution both require the use of the UART pins, this does not leave enough communication capable pins to physically connect the devices. An overall control system input/output (I/O) diagram is shown in Figure 33.

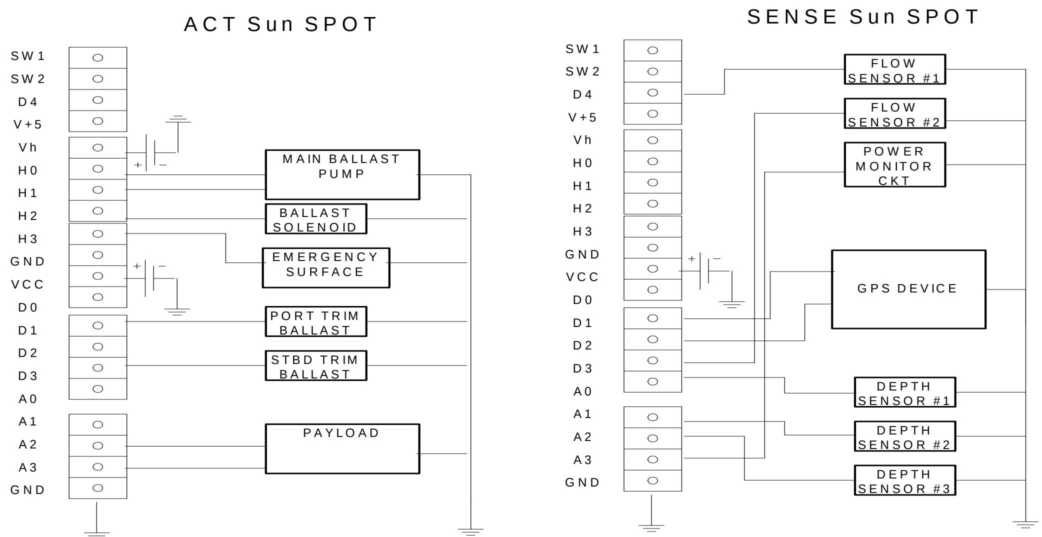


Figure 33. Master I/O diagram for Sun SPOT control system

L. SUMMARY

This chapter described in detail each individual component and system required for successful vehicle operation along with detailed testing and evaluation information for newly acquired systems that were selected for interface with the Sun SPOT.

VI. CONTROL SYSTEM DEVELOPMENT

A. INTRODUCTION

This chapter describes the software development environment, process framework, and controller implementation. It begins with an introduction to the selection of the Sun SPOT for evaluation, a detailed description of the software development environment, then outlines in a sense-decide-act organization, providing the interfaces for each system investigated in chapter V. The chapter concludes with a detailed description of the prototype control system used for bench testing, and an alternative control system device configuration.

B. SUN SPOT AS A CONTROL SYSTEM

In traditional control system development, system requirements are developed and control processes are formulated for programming while being constrained by the actual controller capabilities. Due to the unfamiliarity with the Sun SPOT as a controller, and not having an existing control system to duplicate or replace, system requirements and processes for the vehicle were being developed at the same time that controller capabilities and limitations were being investigated.

The Sun SPOT was chosen for evaluation as a controller due to many of the features that come with the eDemo board. The built in sensors, inherent communication capability, and input output functionality all accessible to the programmer via function calls written in Java are all very attractive features in control system development. As the Sun SPOT was in its infancy when selected for evaluation, there were a number of shortcomings and revisions from Sun Microsystems, as well as a steep learning curve due to the uniqueness of the technology.

C. SUN SPOT SOFTWARE DEVELOPMENT KIT (SDK)

1. Description

The Sun SPOT Software Development Kit (SDK) that was initially distributed with the first release (Green V 1.0) was simply a package of libraries and plug-ins for the NetBeans integrated development environment (IDE). With the second release (Orange

V 2.0), the Spot Manager Console shown in the screenshot in Figure 34 was made available for download. The SPOT manager has been upgraded and revised multiple times, but remains an excellent interface for Sun SPOT devices as well as downloading updates to the documentation and SDK.

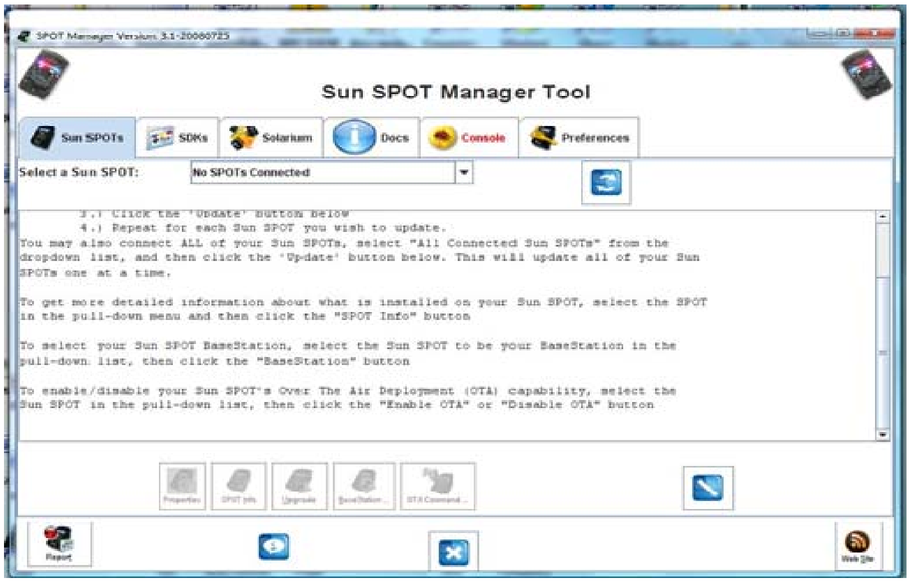


Figure 34. Sun SPOT Manager.

The ability to develop, test, and evaluate operation of the Sun SPOT device using a laptop or personal computer is a major advantage of this approach. As software development continues, advanced reliability engineering can be accomplished using the Java language that is not possible using lower-level processor-specific programming languages.

2. Configuration

As the Sun SPOT was in its infancy when research began, there were a number of short duration upgrades to the SDK. The Sun SPOT Manager Tool above made it much easier to download and install updates, but there were still a number of misalignments and disconnects in the SDK installation process. A number of weeks were spent trying to establish an installation configuration that would satisfactorily enable building and deploying the demonstration code to a Sun SPOT. A final compilation of the installation

process collected from multiple resources and tested is included as Appendix A. The final configuration made operational was the Purple (Version 3) SDK running on the NetBeans 6.1 IDE.

Almost two months were spent trying to configure the IDE and the SDK to even successfully deploy and execute the provided demonstration software. The most basic demonstration piece of code, the “Air Text” demo was the first, and virtually only demo that compiled and executed without issue. Unfortunately this led to a false sense of security about the stability of the development environment.

3. System

The original configuration was only compatible with the Windows XP operating system. Originally there was not driver support for the Sun SPOT to operate on Microsoft Windows Vista. Through trial and error, and collaboration on the Sun SPOT forums, a change to the driver information file, and its location made it possible to continue development using a Dell XPS M1330, running the Windows Vista operating system with no service pack upgrades. The information file modification process is included in the set-up procedure in Appendix A.

4. Challenges

There were a number of challenges just establishing a stable software development environment. Through the first three upgrades to the SDK, there were several items that were not validated before the SDK was released. There were several issues with libraries in the demo code, and in the plug-in for the NetBeans IDE that were not correctly linked. At this point in time, there was no venue for submitting bug reports to Sun with the exception of the Tech Forums. For many of the items reported, fixes were found by independent developers, and Sun acknowledged that they would try and fix the bugs before the next release. A number of problems were encountered that were due to the newly deployed Microsoft Vista operating system.

Another challenge in the beginning stages of evaluating the Sun SPOT for employment was that there were hardware functions that were advertised on the Sun SPOT eDemo board, but the software interfaces had not been written yet. Many of the

software interfaces have been updated, but several of them (such as the UART functionality) still have function bugs and problems awaiting resolution.

D. OVERALL CONTROL STRUCTURE

The basic overall control framework for the system controller is based on a continuous sense-decide-act loop. The sensors are polled continuously collecting information about the vehicles attitude and status. Processes are executed using this information, and based on the next location information, decisions are then made, and any necessary action taken to guide the vehicle towards the next waypoint. For clarity, each block in below in the Figure is described separately. The overall system configuration will be described in more detail later in this chapter. Figure 35 shows the sense-decide-act process the control system was modeled after. As described in chapter 5, the largest of vehicle I/O requirements require the use of two devices. The sense function is performed by one Sun SPOT device, while the decide and act functions are performed on the other.

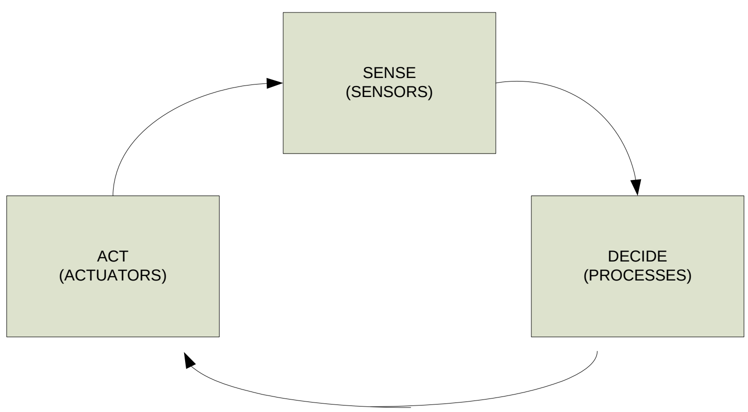


Figure 35. Sense-Decide-Act Process.

E. SENSE

The following sensor data collection framework should all be accomplished on a single Sun SPOT device, and for descriptive purposes will be referred to as the "Sense" SPOT. The sensors must all be polled at regular time intervals and information stored in

a sensor record object. The information collected includes depth, pitch, roll, flow, power status, and acceleration data from the Sun SPOTs built-in 3-axis accelerometer.

This information needs to be kept for at least three iterations to allow for dead-reckoning and delta calculations. At a less-frequent time interval, the data is logged to the recorder for post mission analysis.

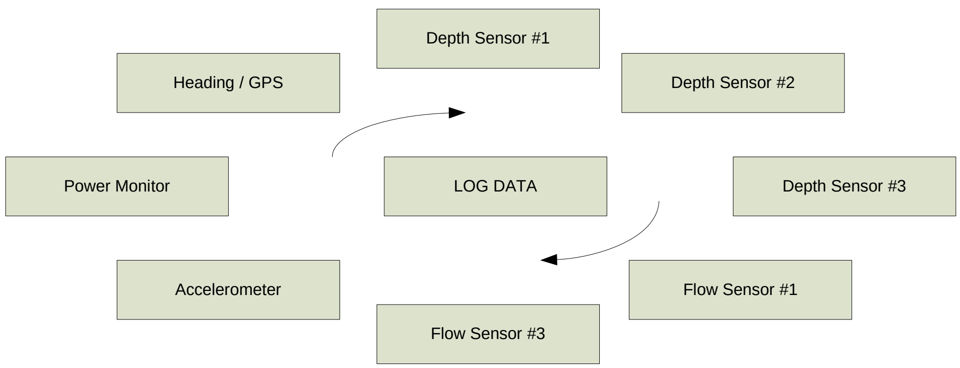


Figure 36. Sense Framework.

Based on the number and types of sensors assumed in Chapter V, a proposed device interconnection diagram is shown below.

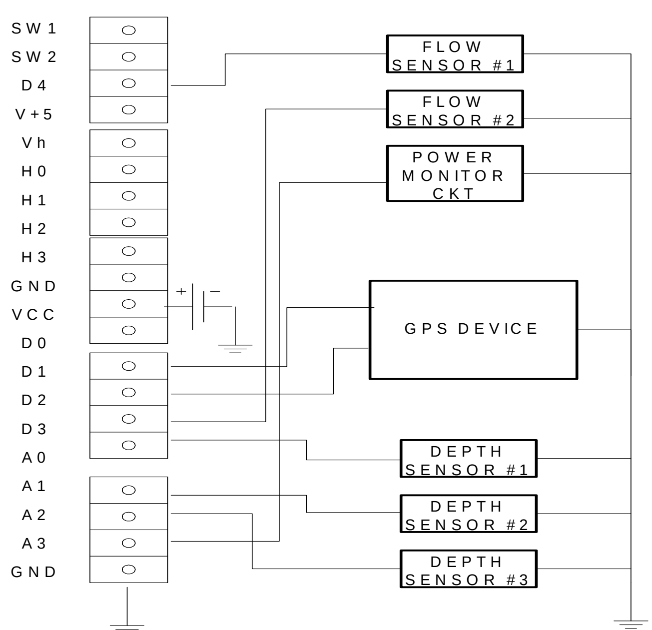


Figure 37. Sense Device Interconnection.

1. Depth

The depth-sensor values are checked by accessing the digital to analog pins on the eDemo board. The pressure from the three sensors described in Chapter V should be stored as three separate values for comparison. The single forward depth-sensor value is necessary for navigation, but can also be compared to the aft two sensors to establish vehicle pitch information, similarly the aft two compared to establish roll as described in Figures 29 and 30 in Chapter V. This same information can also be gathered from the accelerometer inherent to the Sun SPOT, and values compared to establish relative error. As stated in the hardware interface description, the sensors are 5 volt sensors, and the Sun SPOT is limited to 3.3 Volts, so some scaling needs to be taken care of by the hardware voltage divider, in addition to software scale conversions.

2. Accelerometer

Pitch and roll information is gathered by accessing the onboard accelerometer pitch function for the X –axis and Y- axis. The origin and axis layout for the sun SPOT is depicted in the Figure 38 below. The orientation of the X, Y and Z axes follow the right-hand rule (RHR), corresponding to thumb, forefinger, and middle finger respectively. Acceleration information on all three axis must be collected for dead reckoning between waypoints. Velocity is calculated using the acceleration data from the onboard accelerometers, and the time interval. This may require additional gyro input for more accurate operation.

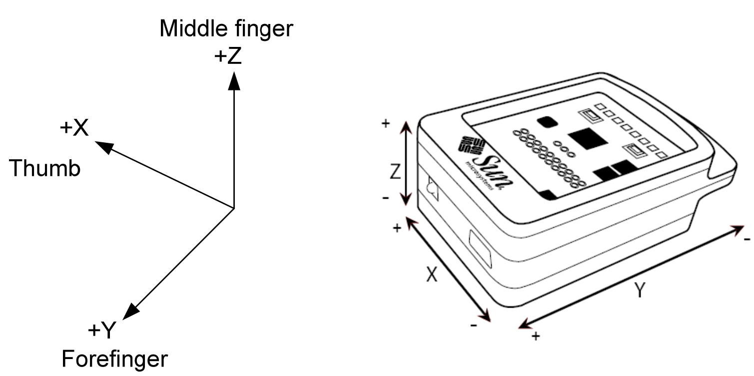


Figure 38. Right-hand rule orientation of X, Y, and Z Accelerometer Axes. (adapted from Sun Microsystems Inc., 2007)

3. Flow Sensors

Two flow sensors will be placed on the top and bottom of the vehicle at its thickest dimension to measure water motion past the hull . This sensor information will be read in via the GPIO interface and compared for vehicle efficiency calculation and correction. The vehicle is operating at optimum efficiency when water flow is equally distributed above and below the vehicle. If the flow over the top of the vehicle is greater than below the vehicle, or vice versa, the desired pitch angle can be adjusted via buoyancy correction. This information can also be used when executing turns, or during dive-cycle transitions, to help improve vehicle dynamic stability.

4. Heading and GPS

The initial heading can be programmed in at vehicle launch, but such prerequisites are difficult to achieve in practice. A suitable electronic compass has not yet been found to interface with the Sun SPOT. Once the heading is established, variations from that initial value will be computed via delta calculations from the accelerometer data. The potential problem with this method, despite claims made by individual developers, the way the Sun SPOT processes the accelerometer data makes it prone to errors. Such reliability of measurement must be corrected.

The initial surface location is ideally be taken by a GPS fix. This information needs to be saved and used for calculations until the GPS signal was lost, the navigation controller can then set a Boolean flag indicating the change in status, and revert to dead-reckoning using the embedded accelerometer sensors.

5. Power

The internal power status of the Sun SPOT is monitored by using the inherent functions provided for in the power controller API. The battery monitor circuit for the main vehicle batteries will provide an input to be accessed through the GPIO pins on the main spot board. Once initial processing requirements have been determined, further control of power consumption may be possible to improve vehicle endurance.

6. Data Logging

As described in Chapter V a suitable external data logging solution is necessary, since the internal memory capacity of the Sun SPOT is not sufficient for the requisite amount of data collected. Due to the limitation of only one available UART set per Sun SPOT device, it is not possible to connect the GPS and a memory storage device to a single device. It is therefore recommended that the sensor data be passed to the “Decide” Sun SPOT device. Since decision making will be its primary function, it is recommended the eDemo board be replaced with the eFlash board described in Chapter V.

It is recommended an object is created for each polling cycle with variables for each sensor value, and a time-stamp and stored in a record class. The initial code framework for this record structure is included in Appendix D, but has not yet been implemented. This information should be passed to the “Decide” Sun SPOT via radio-stream data connection, and stored in an array of at least three records for decision making, as well as permanently logged in the storage device.

F. DECIDE

The decision making component of the system is representative of the Central Processing Unit (CPU) in a traditional consolidated control system. In this control system design, it is implemented on a separate Sun SPOT device, and passes information to and from the “Sense” device, and the “Act” device. The “Decide” device, as in a traditional CPU, is responsible for the initialization process, as well as continuous iterations of the mission execution process as depicted below in Figure 39.

Conceptual processes were developed in the system design phase, but cannot be completely refined into functional controller code until vehicle construction is complete and hardware system parameters established.

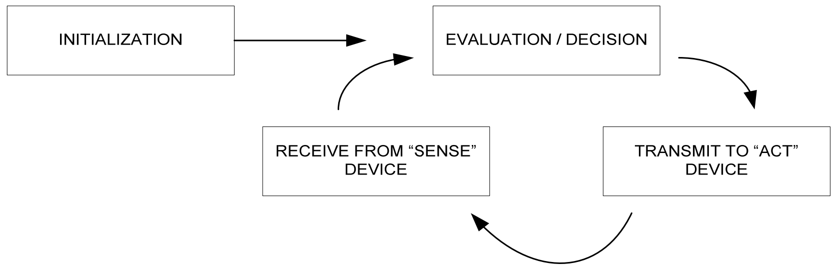


Figure 39. Mission Execution Cycle.

1. Initialization

When the vehicle is initially launched, all systems must be initialized, and a sensor check should be conducted, as well as a communication check between controlling devices and the base station.

In the event there is a discrepancy, the dive mode should immediately be disabled to prevent inadvertent mission execution, the process should be aborted, and error message sent back to the base station. The initial process flow is depicted below in Figure 40.

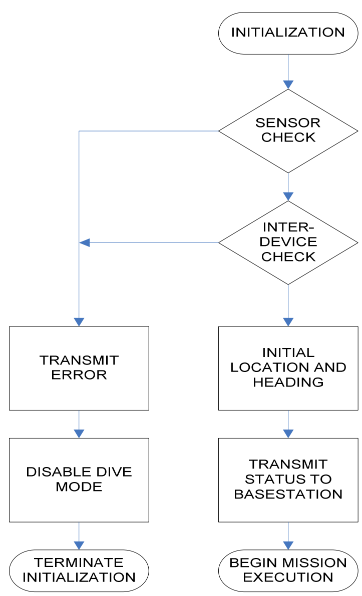


Figure 40. Conceptual controller initialization process flow.

2. Mission Execution

The basic conceptual mission execution process flow is depicted in Figure 41. As with the system initialization, the flow for this process is a simple representation of the

cycle, and each element must be iteratively refined as actual vehicle systems are developed. Once the execution process has been initiated, consistent management of the sense, decide, act loop as well as monitoring for emergency situations is neccessary. Without many of the critical navigation and information systems having been selected and successfully tested, further refinement of these process flows will be necessary as vehicle development continues. It is recommended that basic system level functionality be established, and each tested individually as the vehicle construction and testing cycle progresses.

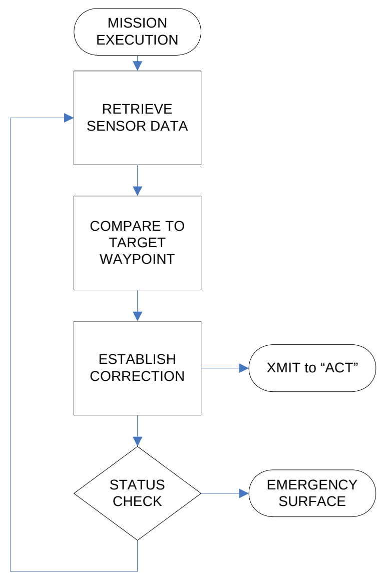


Figure 41. Conceptual basic mission execution flow.

Inherent in UUVs is the possibility that in the event of major system failure, the UUV must inflate the ballast and return to the surface, or else the vehicle is lost. Once an emergency situation from the sensing framework has been detected, a flag is set that overrides all other decision cycles, and the signal is sent to the "Act" device to emergency surface.

G. ACT

The functions in the Act process simply access the most recent correction values received from the "Decide" device, and send the necessary electrical signals to the actuators to perform the required attitude corrections. As this is a continuous loop, the corrections will are checked while they are in progress. This approach prevents sporadic

oscillations and over-correction. This is another area that will require iterative refinement as the vehicle systems are assembled, and final actuator specifications are acquired. The framework for the controller code has been established and is included in Appendix D. The code framework should include the basic inter-device communication interface, and then based on which actuator is connected to which GPIO pin, the appropriate level should be set to high or low accordingly. Based on the number and types of actuators assumed in Chapter V, a proposed device interconnection diagram is shown in Figure 42.

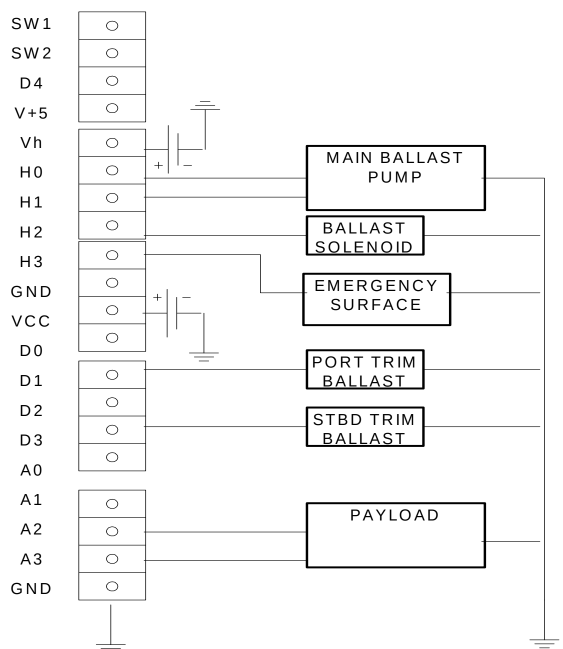


Figure 42. "ACT" Device Interconnection

A significant challenge that was discovered is the need for an electrical interface between the low power outputs of the Sun SPOT and the devices it controls. The proposed devices from Chapter V all operate on 12 volts, and vehicle power from batteries is also 12 volts. One proposed solution might be to research and identify relays that can be activated by 3 volt signal that switches the 12 volt actuation signals in a similar manner to that of a motor-controller. Another solution that may be required is to use electronic logic gates and buffers to step the signal up to 5 volts to actuate relays. These relays are best be implemented on a board in the dry box with the controllers.

H. COMMUNICATION

Due to the limitations of only one UART interface pair per Sun SPOT device, inter-control device communication needs to be established using the inherent wireless radio-stream capability of the Sun SPOT devices. As these devices should all be located in the same dry box compartment, testing may need to be conducted to ensure there is no signal interference due to container construction material. As stated in the communication section of Chapter V, a suitable external communication capability will be required to communicate status to the base station, as well as interface in a testing environment.

I. PROTOTYPE CONTROL SYSTEM CONFIGURATION

The previous sections in this chapter describe an overall control system framework design, and process flow considerations. As in any system design, it is best to begin with basic functionality, and iteratively improve upon, and refine the prototype. In evaluating the suitability of the Sun SPOT as a robot controller a smaller-scale system has been developed that proves the device's ability to read a sensor value, collect vehicle attitude information, actuate an externally connected device, and output data for analysis. The system utilizes two Sun SPOT devices vice the three as described in previous sections of this chapter, a "Sense" device, and an "Act/Decide" device. As the system is still in development, a section outlining alternative configuration considerations follows.

The prototype control system is configured on an electronic prototyping breadboard to provide for reliable solder-less connections and is shown in Figure 43.

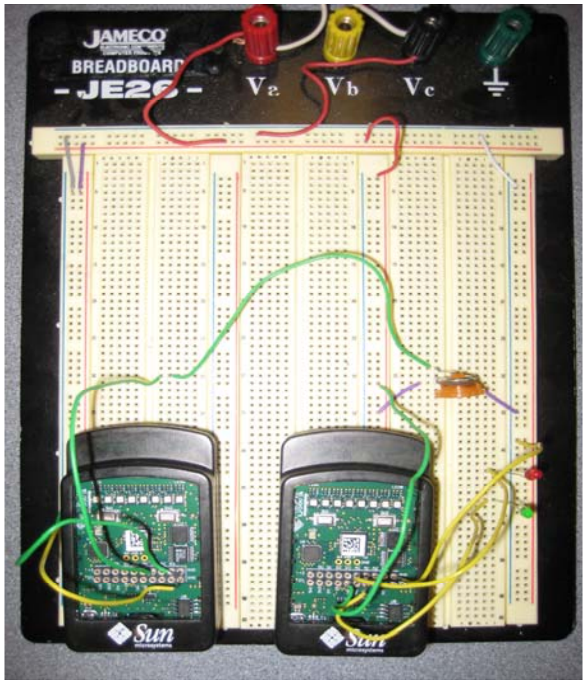


Figure 43. Prototype control system test bench.

The power supply to the test bench is a 6 volt, 800 milliamp power converter. This power source is not connected to the Sun SPOT devices, it simply provides power to the simulated sensor signal, and a stable system ground.

A depth sensor is simulated by connecting a variable potentiometer between the positive lead of the power supply, and system ground and is used to vary the voltage between 0 and 7 volts on the output. This output is connected to pin A1 on the "Sense" Sun SPOT device.

Test bench power is also connected to the Vh pin on the "Act" device to provide the source voltage for the H0 through H3 output pins. The system connection diagram is shown in Figure 44.

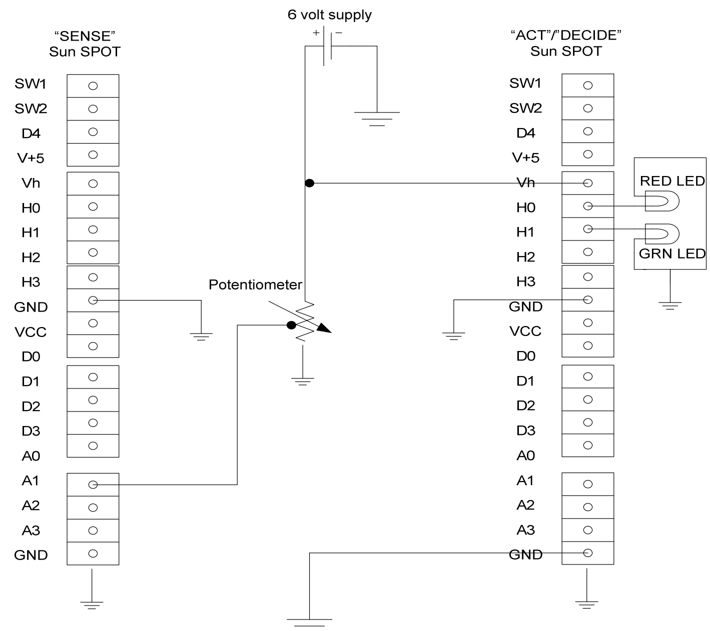


Figure 44. Prototype control system wiring connections.

The "Sense" device reads the accelerometer data, temperature, and voltage applied to pin A1 as if was a connected depth sensor, and transmits that sensor data to the Act/Decide device via wireless datagram. The operational code is included as Appendix B.

The "Act/Decide" device receives the datagram packets, and lights one of the two LEDs to represent actuation of a main ballast pump, or trim ballast pumps as needed to maneuver the vehicle. The LEDs are used to demonstrate the device controlling an actuator. As described in chapter V, many of the pumps and actuators are 12 volt systems, but this 6 volt control signal can be applied to a suitable motor-controller device to control the higher voltage devices.

As sensor data is collected by the "Act/Decide" device as received from the "Sense" device as well as its embedded sensors, it is output to the system console in NetBeans to demonstrate the ability to export the information. Once a suitable data logging device is identified, completing the logger only requires minor additions to the code. An example of the data output is included as Appendix F.

J. ALTERNATIVE CONTROL SYSTEM CONSIDERATIONS

In the early stages of control system development discoveries were made suggesting possible alternative configurations for the number of Sun SPOT devices utilized. The configuration shown in Figure 45 suggests a three device configuration where the sense and decide functionality is combined on a single device (#2), while device number three controls the actuators. The remaining device is connected to the GPS device, and collecting that information is its sole purpose. This configuration allows the navigation Sun SPOT to go into deep sleep mode while the vehicle is submerged. If the power source for the GPS device was supplied from that Sun SPOT device, both devices can essentially hibernate for the majority of the mission duration potentially resulting in power savings that would lengthen mission endurance.

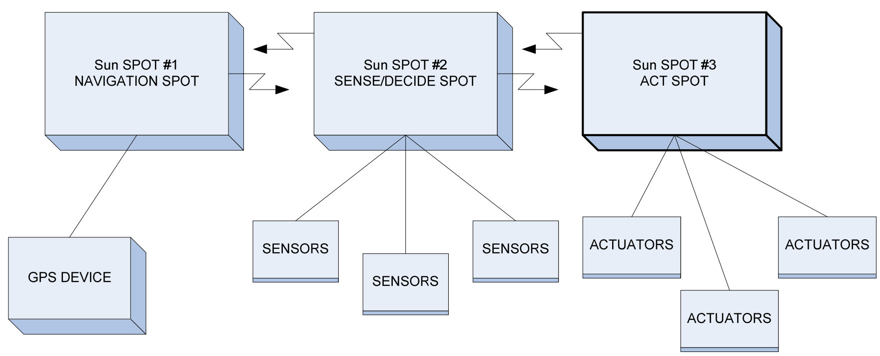


Figure 45. Alternative control system device configuration.

K. SUMMARY

This chapter described the software development environment, process framework, and controller implementation. The first section was an introduction to the selection of the Sun SPOT for evaluation, followed a detailed description of the software development environment. The remainder of the chapter outlines in a sense, decide, act organization, the interface for each system investigated in Chapter V. The chapter concludes with a detailed description of the prototype control system used for bench testing, and an alternative control system device configuration.

THIS PAGE INTENTIONALLY LEFT BLANK

VII. TESTING AND EVALUATION

A. INTRODUCTION

This chapter describes a possible test tank setup designed to thoroughly conduct vehicle testing in the tank environment before deploying the vehicle to sea, followed by a section on preliminary considerations for the division of virtual and actual signals to be interfaced for testing.

B. TEST TANK SETUP

A gliding vehicle presents a number of challenges not faced when testing propeller-driven vehicles. Due to the nature of gliding vehicles, static testing could not be simply conducted in a wet tank without some additional considerations. The first of these considerations is that the vehicle must begin a dive cycle to achieve forward motion, and the depth of the tank is not suitable for this to take place. Therefore, for accurate parameters to be established in relation to vehicle buoyancy, there must be a way to measure the buoyant force allowing accurate prediction and simulation of dive and rise rates, and estimate of vehicle trajectory.

The system depicted below was designed to be attached to the crane beam above the NPS test tank, allowing the vehicle to be tethered in a neutrally buoyant state approximately 6 to 8 inches below the water surface. The vehicle is suspended from the overhead using plastic coated 1/8th inch wire rope with a 250 pound load rating. The two force meters were inserted to measure the force of buoyancy during dive and rise cycles. As the meters will be utilized in a wet environment, the initial system employed two scales designed to weigh fish. The scales are connected to the cables with eyelets and loopholes in the cable, and an inline type cable tension adjuster was placed in each section of the forward measurement cable. The aft cable is a single piece that spans from the ceiling to a weight to be placed at the bottom of the tank, and connects to the vehicle through a screw in eyelet that allows the vehicle vertical motion in the water, but prevents forward and aft motion, as well as restricts vehicle heading changes.

All cables are connected to the vehicle at the centerline of the body, with the forward cables being placed at the center of buoyancy. The cable and force meter system has been constructed and remains available in the NPS Unmanned Vehicle Systems Lab and is shown in Figure 46. The Sun SPOT from glider transmits recorded results to base station. Weight scales display measured changes in buoyant force.

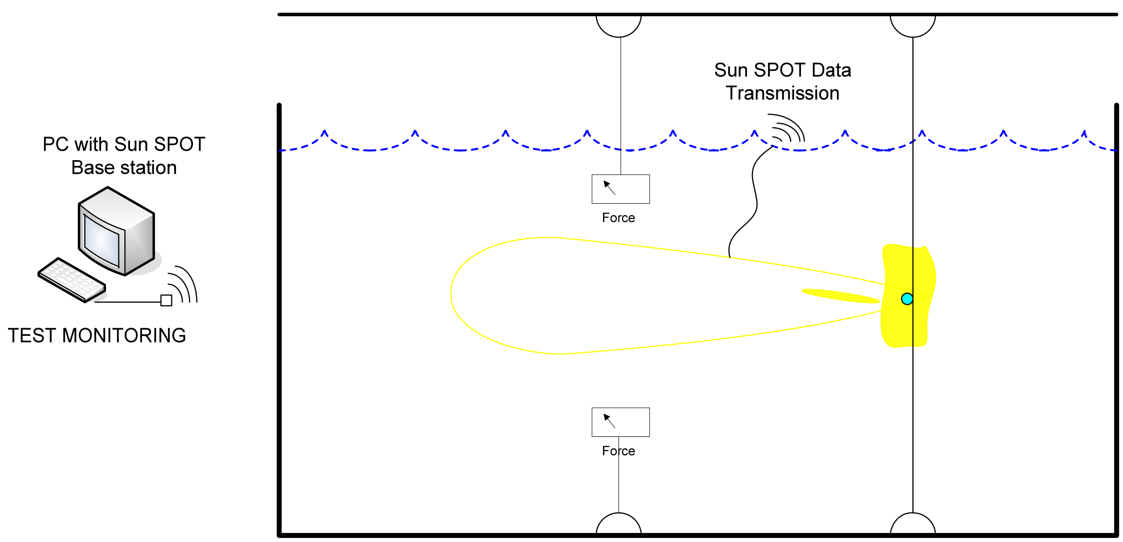


Figure 46. Notional test tank design.

C. SENSORS AND SIGNALS

All systems are intended to operate as on a regular mission, though some sensors without appropriate inputs need to be overridden by virtual signals. Based on system characteristics as described in Chapters V and VI, a preliminary evaluation was conducted into which actual signals can be utilized, and which virtual signals need to be injected (or substituted) in the simulation software.

1. Virtual Signals

As the vehicle will never actually change depth, and forward motion will be restricted in the test tank environment the following virtual signals need to be substituted for actual:

- Depth Sensors
- Flow Sensors

2. Actual Signals

The following actual signals are expected to be suitable for testing purposes and require communication of these values from the controller to the simulation software:

- Pitch and Roll
- Buoyant Force (from meters)
- Battery Level
- Buoyancy actuators

D. VEHICLE INTERFACE

Care should be taken when introducing test-interface code and connections directly to the Sun SPOT. As described in Chapters V and VI, once a reliable means to communicate externally to the vehicle has been established, the controller can be accessed wirelessly via the base station. This type of interface is thought to be ideal, as external connectors to the control box are not necessary, and it supports the need for development of the software interface between AUV Workbench and the Sun SPOT for functional real-time operational requirements. A recommendation for future research and development is to develop functionality in AUVW that controls communication directly to the Sun SPOT via the base station, completely closing the testing loop.

E. DATA COLLECTION AND ANALYSIS

As vehicle and control system development were in progress, report generation functionality was created and added to AUVW. AUVW already had the capability to take post mission results that had been converted to AVCL and replay the missions. The report generator employs and XSLT style sheet to parse vehicle and planning data out the pre-mission set up file, and also parses the mission results and converts those to an HTML report for web based review. The report generator is constantly being refined to include additional mission information, as well as 3D replay capability using X3D graphics.

Not only might this be a powerful tool for review and analysis of test results, but as the vehicle communication system is refined, this might enable real-time review, replay, and possibly real-time control remotely via The Internet.

F. SUMMARY

This chapter describes a possible test tank setup that was developed to thoroughly conduct vehicle testing in the tank environment before deploying the vehicle to sea, followed by a section on preliminary considerations for the division of virtual and actual signals to be interfaced for testing. The chapter concludes with recommendations for interfacing with the vehicle, as well as considerations for testing and data analysis

VIII. CONCLUSIONS AND RECOMMENDATIONS

A. CONCLUSIONS

This research has examined and explained in detail the theory behind the vehicle and control system development of a free flood, fully scalable, low cost, long-range gliding underwater vehicle. Conclusions were drawn in three main areas.

1. SeaDiver II Gliding Vehicle Development

The main motivation for this research was to evaluate the Sun SPOT as a suitable controller for the physical model to be integrated with simulation software to conduct loop testing. The research began by evaluating previous work, and continuing development of that model. Through research conducted by the team at ISITV, the vehicle prototype provided many lessons learned, and items to be taken into consideration as vehicle development continues. There were a number of logistic issues in recovering the vehicle from the French team, which caused significant delays. This thesis outlines the vehicle in enough detail that a new vehicle might be created utilizing locally obtainable parts. Through this research it can be concluded that a vehicle design of this type warrants further development and refinement.

2. Sun SPOT Development Environment

The biggest challenge to conducting this research was investigating a new technology that underwent four major software development kit, and interface revisions during an eighteen month timeframe. As the upgrades were released, there were a number of areas were due attention may not have been paid to documentation and interface integration. Through this research, it has been discovered that the developmental instability will remain a challenge until a suitable development environment is established. The steady delivery of revisions and an active development community remain promising assets for supporting continued work.

3. Control System Development Utilizing Sun SPOT Technology

Control system development in parallel with physical vehicle development is a challenging process even when the capabilities and operating parameters of the control system are known. Investigating a new controller technology while simultaneously developing vehicle systems, and the control framework in this research may not have been the most effective or efficient approach. Challenges faced in the software development directly contributed to a number of unsuccessful attempts to continue vehicle system development to the inability to interface with devices external to the Sun SPOT. It cannot be concluded that the Sun SPOT is an unsuitable option, however the developmental instability must be overcome before successful development can continue. Further testing, experience and software engineering is likely to help

B. RECOMMENDATIONS FOR FUTURE WORK

1. Finish Construction of SeaDiver II

As stated in the conclusion above, the actual vehicle and subsystems must be successfully delivered, or construction of a new model must be completed before control system development can continue. An itemized listing of parts on hand, and parts still required are included as Appendices G and H respectively.

2. Continue Control System Development and Evaluation

Once a complete vehicle, or subsystems thereof have been constructed, control system development and evaluation can continue. This thesis provides the framework based on the progress made on vehicle development.

3. Develop an External Communication Solution for Sun SPOT

Although the Sun SPOT documentation states that Federal Communication Commission (FCC) constraints prohibit the addition of an external antenna to the device,

this area requires further research and development to ensure wireless communication between the vehicle and the base station. Of note is that U.S, maritime, and international restrictions each differ.

4. Develop Sun SPOT Interface Functionality In AUV Workbench

It is recommended that once a stable control system has been developed, device communication and control functionality be integrated directly into the AUVW software suite.

As this control system was being evaluated for the SeaDiver II gliding vehicle, the open source hardware and software of the control system, coupled with an open source planning and control suite might significantly impact the future of unmanned vehicle development.

5. Integrate Performance Data Into Previously Developed Simulations

Once testing has been conducted and iterations of vehicle refinement continue, any vehicle parametric data discovered needs to be periodically entered into the mentioned simulations that have already been developed to continually check feasibility and continued system improvement.

6. Conduct At Sea Testing of SeaDiver II

This thesis described a robust test tank solution that should enable the vehicle to run a full mission in the test tank while continuously collecting performance data for refinement. Once a full successful mission has been executed in the testing environment, the SeaDiver II should be iterated through a series of missions with increasing complexity.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A. SDK CONFIGURATION

Sun SPOT Development Environment Download and Set-up

****VISTA NOTE:** Turn off User Account Control to allow SDK installer to work properly: Start -> Control Panel -> User Accounts -> User Accounts -> Turn User Account Control on or off , reboot. (you can turn it back on if required after the sdk is installed)

****Before Beginning:** Recommend you check your java installs and ensure you have only one copy of the latest jdk and jre installed. If this is not the case, use the UNINSTALL feature in control panel to clean directories.

1. Download and Install NetBeans IDE 6.1

http://www.netbeans.org/downloads/

**Recommend downloading the latest Netbeans / JDK combo package.

2. Installing Sun SPOT plugin

Install Sun SPOT plugin into NetBeans. Download the Update Center from: http://www.sunspotworld.com/NB6/com-sun-sunspot-updatecenter.nbm

This will save as a .zip file, so as you save, rename to .nbm. Now, in NetBeans, go to:

1. Tools menu -> *Plugin*.
2. Click on *Downloaded* tab.
3. Click on *Add Plugins* button and select the file. Then, confirm the installation clicking on *Install* button and accept the license agreement.
4. Finally, go to the *Available Plugins* tab, sort by category, and mark SunSPOTApplicationTemplate, SunSPOTHostApplicationTemplate and Sun SPOT Info, click on *Install* button and accept the license agreement.
5. Go to settings tab, select it to verify update center is installed.
6. Go to available plug-ins tab and reload catalog.
7. Sort by category
8. You can now check that everything is installed going to *Installed* tab.

3. Getting Sun SPOT SDK 3.0 (Purple Version)

1. If you are not sure you have Ant installed on your system, download it from http://ant.apache.org/ and install it (unzip it to a directory named C:\ant).

2. Check and Set all the environment variables, to do this, Start -> Control Panel -> Enter "env" in the search box (no quotes) -> click "Edit the system environment variables". Click "Environment Variables, now in the upper section (user variables):

New -> name: ANT_HOME, value: Path to your ant installation, say C:\ant
New -> name: JAVA_HOME, value: Path to your java installation, say C:\Program Files\Java\jdk1.6.0_03

EDIT the path variable so it includes the path to the bin directory of your Ant installation (for example: C:\SSH Secure Shell;C:\ant\bin) (use ; to separate)

<u>Ensure ANT is in your root directory, as this is the only place the system checker will look.**</u>**

3. Perform a complete system RESTART.

4. the Sun SPOT SDK. Go to http://www.sunspotworld.com/SPOTManager/ Be sure you have at least JDK 1.5 and at least Ant 1.6.5 installed and configured in your system. If the system checker does not find your installation of Netbeans, that is fine, it is not necessary to pass the system check.

*****VISTA NOTE**Before the first time you plug a SPOT in, follow this: Your OS will not recognize and automatically install the drivers. Do this manually by pointing the system directly to the .inf file in the SDK directory, NOT the Windows.inf folder!***

Check SunSPOT.inf in your SDK directory. Make sure it contains the line "include=mdmcpq.inf" so it looks as follows:

[SpotInstall]
include=mdmcpq.inf
CopyFiles=SpotCopyFiles
AddReg=SpotAddReg

4. Spot Manager will do a system check, and download the latest SDK.

4. Updating Demos And Sunspots

1. Open the Spot Manager using the icon that was installed on your desktop.

2. Click on the preferences tab.

3. Select the “Beta Update Center” radio button.

4. Click the SDK tab.

5. Click on the "Demos" Icon at bottom right. This will download the latest demos.

6. Verify that the latest demos to be installed, match the SDK you have installed (Purple)

7. This will create a folder called "Demos1" in your SDK directory (typically C:\Program files\sun\Sunspot\SDK) This is because there already existed a Demos folder. Delete the Demos folder, and rename the new Demos1 folder to just read Demos. This will insure the links from the SunSpot console in NetBeans link to the most current Demos.

Creating the Demo Application

To make the default demo application work. In NETBEANS Go to:

1. *File* menu -> *New Project*
2. Select *Java* category, select *Sun SPOT* project and then click *Next*.
3. Leave the default project name and package and click *Finish*.
4. Open org.sunspotworld.StartApplication.java. You can see it extends MIDlet and already has some code. If you are familiar with Java ME, you will not have big problems. But, basically, Sun SPOT architecture is CLDC (Connected Limited Device Profile) 1.1 and has IMP (Information Module Profile) in the top - which can be defined as a MIDP (Mobile Information Device Profile) without UI stuff. Also, it has some additional libraries and all this runs in a VM called Squawk, that is characterized by being most written in Java. So, StartApplication implements the abstract methods inherited from the MIDlet (startApp,pauseApp that Squawk never calls and destroyApp). It works in a sand box enviromnent.
**You are likely to encounter an issue with the radio datagram library import if so ensure your classpath in properties look as follows:
C:/Program Files/Sun/SunSPOT/sdk/lib/squawk_rt.jar;
C:/Program Files/Sun/SunSPOT/sdk/lib/spotlib_host.jar;
C:/Program Files/Sun/SunSPOT/sdk/lib/spotlib_common.jar;
C:/Program Files/Sun/SunSPOT/sdk/lib/multihoplib_rt.jar;
C:/Program Files/Sun/SunSPOT/sdk/lib/transducerlib_rt.jar;

The startApp code basically gets an object reference to the singleton eDemo Board and then make its LED blink red for a quarter of second each second. As you, can see, the code is pretty high level and easy to understand.

```
    protected void startApp() throws MIDletStateChangeException {

        System.out.println("Hello, world");

        new BootloaderListener().start();   // monitor the USB (if connected) and recognize commands from
host

        long ourAddr = Spot.getInstance().getRadioPolicyManager().getIEEEAddress();

        System.out.println("Our radio address = " + IEEEAddress.toDottedHex(ourAddr));
```

```
        ISwitch sw1 = EDemoBoard.getInstance().getSwitches()[EDemoBoard.SW1];
        leds[0].setRGB(100,0,0);                // set color to moderate red
        while (sw1.isOpen()) {                  // done when switch is pressed
            leds[0].setOn();                    // Blink LED
            Utils.sleep(250);                   // wait 1/4 seconds
            leds[0].setOff();
            Utils.sleep(1000);                  // wait 1 second
        }
        notifyDestroyed();                      // cause the MIDlet to exit
    }
```

Running the Demo Application in the emulator

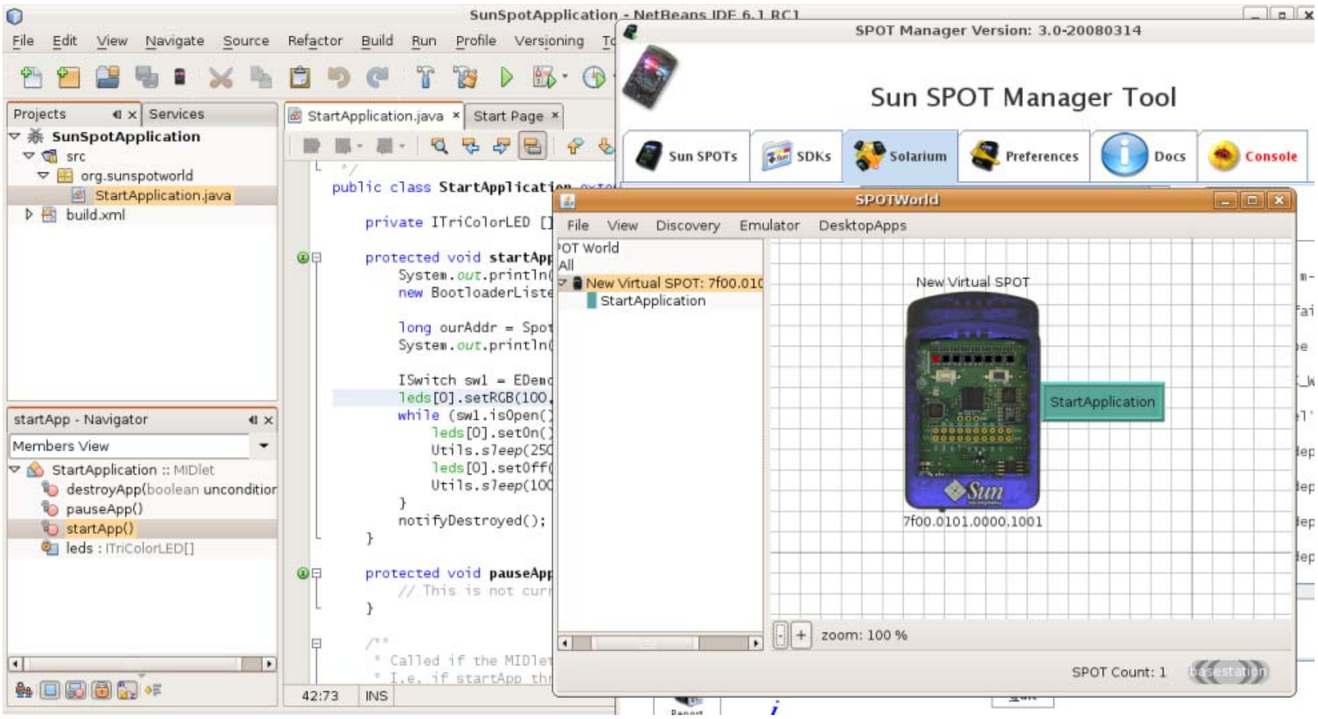
Put this application to run in the emulator.

1. Right click the project and select *Build*. It will generate the jar in the /$PROJECT_HOME/suite/. If you have a spot device, you could send it directly by right clicking the project and selecting *Build Project + Deploy to Sun SPOT* , forgetting the next steps.

2. Open the emulator (in the ToolManager, go to *Solarium* tab and then click in the *Solarium* button). Then, in the emulator, click on the *Emulator* menu -> *New virtual SPOT*, you will notice that a Sun SPOT will appear in the squared right area.

3. Right click on the Sun SPOT picture, then click *Specify application jar file...* and select the application jar in /$PROJECT_HOME/suite/, which is the place that the application was built in step 5.1.

4 . After that, click again on it, then *Run MIDlet* and *StartApplication*. Finally, you will see a red LED blinking.



REFERENCES:

SunSPOT Help Forum. https://www.sunspotworld.com/forums/viewtopic.php?p=5940

Bruno Ghisi's Blog.
http://weblogs.java.net/blog/brunogh/archive/community_netbeans/index.html

David Simmons' Blog. http://blogs.sun.com/davidgs/entry/netbeans_6_0_and_sun

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B. SENSE DEVICE CODE

```
/*
 * @author: Ron Hemmelgarn
 * @date: 19 September
 * @file: seaDiverSenseController
 *.
 *The purpose of this code is to act as the sense controller portion
 * of a two part control system being developed to control the SeaDiverII
 * gliding underwater vehcile being developed at the Naval Postgraduate School
 * in Monterey California.
 *
 * This code was adapted and modified from portions of the Sun SPOT demo
 * code as well as includes adaptaions made by Chris Fitzpatrick and Ken
 *Maroon.
  *
 *Copyright (c) 1995-2008 held by the author(s).  All rights reserved.

 *Redistribution and use in source and binary forms, with or without
 *modification, are permitted provided that the following conditions
 *are met:

     * Redistributions of source code must retain the above copyright
       notice, this list of conditions and the following disclaimer.
     * Redistributions in binary form must reproduce the above copyright
       notice, this list of conditions and the following disclaimer
       in the documentation and/or other materials provided with the
       distribution.
     * Neither the names of the Naval Postgraduate School (NPS)
       Modeling Virtual Environments and Simulation (MOVES) Institute
       (http://www.nps.edu and http://www.MovesInstitute.org)
       nor the names of its contributors may be used to endorse or
       promote products derived from this software without specific
       prior written permission.

 *THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 *"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 *LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
 *FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE
 *COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
 *INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING,
 *BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 *LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
 *CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 *LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
 *ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 *POSSIBILITY OF SUCH DAMAGE.

  **********************************************************************


 * Copyright (c) 2007 Sun Microsystems, Inc.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 * of this software and associated documentation files (the "Software"), to
 * deal in the Software without restriction, including without limitation the
 * rights to use, copy, modify, merge, publish, distribute, sublicense, and/or
 * sell copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
```

```
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
 * FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
 * DEALINGS IN THE SOFTWARE.
 **/
      package glider;


      import com.sun.spot.sensorboard.peripheral.IAccelerometer3D;
      import com.sun.spot.sensorboard.peripheral.ISwitch;
      import com.sun.spot.sensorboard.peripheral.ITriColorLED;
      import com.sun.spot.sensorboard.peripheral.LEDColor;
      import com.sun.spot.sensorboard.EDemoBoard;
      import java.io.*;
      import javax.microedition.io.Connector;
      import javax.microedition.io.Datagram;
      import javax.microedition.io.DatagramConnection;
      import javax.microedition.midlet.MIDletStateChangeException;
      import com.sun.spot.sensorboard.io.IScalarInput;
      import com.sun.spot.io.j2me.radiogram.RadiogramConnection;
      import com.sun.spot.util.Utils;


public class seaDiverSenseController extends javax.microedition.midlet.MIDlet {

          private ISwitch sw1,  sw2;
          public ITriColorLED[] leds;
          RadiogramConnection conn;
          IAccelerometer3D accelerometer;
          EDemoBoard eDemo;
          boolean sw1IsClosed, sw2IsClosed;
          int sendStatus = 0;  // Send Status 0 = Don't send and 1 = Send
          int controlMotor = 2;
          int accelerationLeftRight;
          int accelerationForwardAft;
          double depthValue;

          protected void startApp() throws MIDletStateChangeException {

   //Lights the 1st four Led's on the top of the SPOT to indicate initialization
              leds = EDemoBoard.getInstance().getLEDs();
              for(int i=0;i<4;i++)
              {
                  leds[i].setOn();
                  leds[i].setColor(LEDColor.RED);
              }
 //Lights the lasst four Led's on the top of the SPOT to indicate initialization
              for(int i=4;i<8;i++)
              {
                  leds[i].setOn();
                  leds[i].setColor(LEDColor.GREEN);
              }
//Initialze the accelerometer
              accelerometer = EDemoBoard.getInstance().getAccelerometer();
              accelerationForwardAft = 0;
              accelerationLeftRight = 0;

//Start the sense and send cycle.  This is where the sensors will be polled
//continuously
//and their values broadcast to the decide/act controller.

             startSenseAndSendThread();

          }

/**
* The Sense and Send thread polls the sensors and sends the data each 500 ms
*/
          synchronized public void startSenseAndSendThread() {
              new Thread() {
                  public void run() {
                      // Create the wireless DatagramConnection
                      DatagramConnection dgConnection = null;
```

```
                    Datagram dg = null;

             try {
// The Connection is a broadcast so we specify it in the creation string
             dgConnection = (DatagramConnection)
             Connector.open("radiogram://broadcast:37");
// Then, we ask for a datagram with the maximum size allowed

             dg = dgConnection.newDatagram(dgConnection.getMaximumLength());
                          }

             catch (IOException ex)
             {
            System.out.println("Could not open radiogram broadcastConnection");
            ex.printStackTrace();
            return;
             }

                    while(true){
                        try {
//Reset the datagram connection
                            dg.reset();
//Get an instance of A/D input pin A1 "SIMULATED DEPTH SENSOR"
                    IScalarInput pinA1 =
                    EDemoBoard.getInstance().getScalarInputs()[EDemoBoard.A1];
 //write the reading to the datagram
                    dg.writeInt(pinA1.getValue()+1);
//Get and write the accelerometer values to the datagram
                  dg.writeUTF(""+accelerometer.getTiltY()+ ":"+
                    accelerometer.getTiltX()+":"+accelerometer.getTiltZ() );
//Send the datagram
                            dgConnection.send(dg);


                        } catch (IOException ex) {
                            ex.printStackTrace();
                        }
                        Utils.sleep(500);
                    }
                }
              }.start();
          }

          protected void pauseApp() {
          }

    /**
     * Called if the MIDlet is terminated by the system.
     * I.e. if startApp throws any exception other than
     *MIDletStateChangeException,
     * if the isolate running the MIDlet is killed with Isolate.exit(), or
     * if VM.stopVM() is called.
     *
     * It is not called if MIDlet.notifyDestroyed() was called.
     *
     * @param unconditional If true when this method is called, the MIDlet must
     *    cleanup and release all resources. If false the MIDlet may throw
     *    MIDletStateChangeException  to indicate it does not want to be
     *destroyed
     *    at this time.
     */
         protected void destroyApp(boolean unconditional) throws
             MIDletStateChangeException {
          }

      }
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX C. ACT/DECIDE CODE

```
/*
 * @author: Ron Hemmelgarn
 * @date: 19 September
 * @file: seaDiverActController
 *.
 *The purpose of this code is to act as the act controller portion
 * of a two part control system being developed to control the SeaDiverII
 * gliding underwater vehcile being developed at the Naval Postgraduate School
 * in Monterey California.
 *
 * This code was adapted and modified from portions of the Sun SPOT demo
 * code as well as includes adaptaions made by Chris Fitzpatrick and Ken Maroon.
 * Copyright (c) 2007 Sun Microsystems, Inc.
 *
 ***********************************************************************
 *
 *Copyright (c) 1995-2008 held by the author(s).  All rights reserved.

 *Redistribution and use in source and binary forms, with or without
 *modification, are permitted provided that the following conditions
 *are met:

     * Redistributions of source code must retain the above copyright
       notice, this list of conditions and the following disclaimer.
     * Redistributions in binary form must reproduce the above copyright
       notice, this list of conditions and the following disclaimer
       in the documentation and/or other materials provided with the
       distribution.
     * Neither the names of the Naval Postgraduate School (NPS)
       Modeling Virtual Environments and Simulation (MOVES) Institute
       (http://www.nps.edu and http://www.MovesInstitute.org)
       nor the names of its contributors may be used to endorse or
       promote products derived from this software without specific
       prior written permission.

 *THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 *"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 *LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
 *FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE
 *COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
 *INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING,
 *BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 *LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
 *CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 *LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
 *ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 *POSSIBILITY OF SUCH DAMAGE.

  ***********************************************************************

 * Permission is hereby granted, free of charge, to any person obtaining a copy
 * of this software and associated documentation files (the "Software"), to
 * deal in the Software without restriction, including without limitation the
 * rights to use, copy, modify, merge, publish, distribute, sublicense, and/or
 * sell copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
 * FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
```

```
 * DEALINGS IN THE SOFTWARE.
 **/
package glider;

import java.util.Date;
import com.sun.spot.sensorboard.peripheral.IAccelerometer3D;
import com.sun.spot.sensorboard.EDemoBoard;
import com.sun.spot.sensorboard.peripheral.ISwitch;
import com.sun.spot.sensorboard.peripheral.ITriColorLED;
import com.sun.spot.sensorboard.peripheral.LEDColor;
import com.sun.spot.peripheral.IPowerController;
import com.sun.spot.sensorboard.io.IOutputPin;
import com.sun.spot.peripheral.Spot;
import java.io.*;
import javax.microedition.io.Connector;
import javax.microedition.io.Datagram;
import javax.microedition.io.DatagramConnection;
import javax.microedition.midlet.MIDletStateChangeException;
import com.sun.spot.sensorboard.peripheral.ITemperatureInput;
import com.sun.spot.io.j2me.radiogram.RadiogramConnection;
import com.sun.spot.util.Utils;


/**This class acts as the decide/act portion of the controller.  The Sensor data
is received
 * from the sense device via wireless radiodatagram.  The send fucntion has been
left in for
 * if and when a third controller is added.
 **/


public class seaDiverActController extends javax.microedition.midlet.MIDlet {


    public ITriColorLED[] leds;
    RadiogramConnection conn;
    IAccelerometer3D accelerometer;
    boolean sw1IsClosed, sw2IsClosed;
    int sendStatus = 0;  // Send Status 0 = Don't send and 1 = Send
     public IOutputPin[] outPins ;
    int controlMotor = 2;
    int accelerationLeftRight;
    int accelerationForeAft;
    private IPowerController powerController;
    public boolean missionStartFlag = false;
    public ITemperatureInput thermometer =
EDemoBoard.getInstance().getADCTemperature();
    protected void startApp() throws MIDletStateChangeException {

        System.out.println("Sample System Output File");
        leds = EDemoBoard.getInstance().getLEDs();
//Start Mission
        missionStartFlag = true;
//Get instance of the output pins
        outPins = EDemoBoard.getInstance().getOutputPins();

        accelerometer = EDemoBoard.getInstance().getAccelerometer();
        accelerationForeAft = 0;
        accelerationLeftRight = 0;
       powerController = Spot.getInstance().getPowerController();
        startReceiverThread();

    }

    /**
     * The receiver thread blocks on the receive function
     * so you don't have to sleep between each receive.
     */
    public void startReceiverThread() {
        new Thread() {

            public void run() {
```

```
                String tmp = null;
                double startTime = 0.0;
                double elapsedTime = 0.0;
                 double vbatt =  0.0;



                RadiogramConnection dgConnection = null;
                Datagram dg = null;

                try {
                    dgConnection = (RadiogramConnection)
Connector.open("radiogram://:37");
                    // Then, we ask for a datagram with the maximum size allowed
                    dg =
dgConnection.newDatagram(dgConnection.getMaximumLength());
                } catch (IOException e) {
                    System.out.println("Could not open radiogram receiver
connection");
                    e.printStackTrace();
                    return;
                }

                while (true) {
                    try {

 /******************************************************************
  * THIS PORTION OF THE CONTROL CODE IS RECEIVING SENSE DATA
  * FROM THE SENSE CONTROLLER VIA WIRELESS DATA GRAM AND WRITING
  * IT TO THE OUTPUT.  CURRENTLY IT IS TO SYSTEM CONSOLE, WRITE
  * STATMENTS TO BE MODIFIED WHEN A RECORD CLASS IS CREATED FOR
  * WRITING TO AN EXTERNAL MEMOR DEVICE.
  *****************************************************************
  */
                        dg.reset();
                        dgConnection.receive(dg);
                        char colon = 58;
                        int heading = 0;
                        int depth = dg.readInt();
                        tmp = dg.readUTF();
//TEST// System.out.println("Received: " + tmp + " from " + dg.getAddress());
                        String[] accelinfo = Utils.split(tmp, colon);
//TEST//System.out.println(accelinfo[0]);
//Parse the accelerometer   info from the datagram
                        double tily = Double.parseDouble(accelinfo[0]);
                        double tilx = Double.parseDouble(accelinfo[1]);
                        double tilz = Double.parseDouble(accelinfo[2]);



                        // Returns [-90, 90], Convert angle to range [-3, 3]
                        int tiltY = (int) Math.toDegrees(tily);
                        int tiltX = (int) Math.toDegrees(tilx);
                        int tiltz = (int) Math.toDegrees(tilz);

//Get heading information and estimate based on tilt value from Z axis
                        heading = heading - tiltz;
                        if(heading<0){heading = 360 - heading;}
                        if(heading>360){heading = heading -360;}
/************************************************
 * Write the output to the system screen in lieu of a data file
 *
 *************************************************/
                        System.out.print("/Time:"+ new
                        Date(powerController.getTime()));
                        //System.out.println("Tilt x:"+tilx);
                        System.out.print("/Depth:"+depth);
                        System.out.print("/PITCH:"+tiltY);
                        System.out.print("/ROLL:"+tiltX);
                        System.out.print("/Heading:"+heading);
                        System.out.print("/Temp:"+thermometer.getFahrenheit());
```

```
                        System.out.println("/Power:"
+powerController.getVbatt()+"mv");
/*************************************************
 *THE FOLLOWING SECTION SETS THE LEDS ON THE SPOT
 TO GIVE THE INDICATIONS THAT THE DATA IS BEING
 RECEIVED CORRECTLY.  FOR TESTING AND EVALUATION
 PURPOSES ONLY
 ***************************************************/
                        int accelerationForeAft = -tiltY / 15;
                       // Set max forward acceleration to -90 degrees
                        if (accelerationForeAft < -3) {
                            accelerationForeAft = -3;

                        }

                        // Set max reverse acceleration to 90 degrees
                        if (accelerationForeAft > 3) {
                            accelerationForeAft = 3;

                        }


                        leds[3 + accelerationForeAft].setColor(LEDColor.GREEN);
// Set Speed/FWD & REV leds to GREEN
                        leds[3 + accelerationForeAft].setOn();
// Turn on GREEN leds
                        leds[4 + accelerationForeAft].setColor(LEDColor.GREEN);
// Additional led lit to easily view fwd/rev movement
                        leds[4 + accelerationForeAft].setOn();
                        Utils.sleep(50);                              // update
20 times per second
                        leds[3 + accelerationForeAft].setOff();
// Clear display
                        leds[4 + accelerationForeAft].setOff();



                        // Returns [-90, 90], Convert angle to range [-3, 3]
                        //int tiltX = (int) Math.toDegrees(tilx);
                        int accelerationLeftRight = -tiltX / 15;

                        // Set max turn direction to 90 degrees rotation left
                        if (accelerationLeftRight < -3) {
                            accelerationLeftRight = -3;
                        }

                        // Set max turn direction to 90 degrees rotation right
                        if (accelerationLeftRight > 3) {
                            accelerationLeftRight = 3;
                        }

                        leds[3 + accelerationLeftRight].setColor(LEDColor.RED);
// Set direction leds to RED
                        leds[3 + accelerationLeftRight].setOn();
// Turn on RED leds
                        leds[4 + accelerationLeftRight].setColor(LEDColor.RED);
// Additional led lit to easily view direction
                        leds[4 + accelerationLeftRight].setOn();
                        Utils.sleep(50);                            // Update
                        20 times per second
                        leds[3 + accelerationLeftRight].setOff();
// Clear display
                        leds[4 + accelerationLeftRight].setOff();
/********************************************
 * DECIDE WHAT TO DO WITH THE SENSOR INPUT
 * BEING RECEIVED AND TAKE ACTION BY STARTING
 * OR STOPPING BALLAST PUMP
 *
 ********************************************/


                        // Glider is level and stable
```

```
                    if (tiltY <-8 && tiltY < 8) {
                    //if beginning of mission start ballast pump
                        if(missionStartFlag)
                        {outPins[EDemoBoard.H0].setHigh();
                        outPins[EDemoBoard.H1].setLow();
                        missionStartFlag = false;
                        System.out.println("Glider is Level and Stable and
                         mission starting ballast on");
                        }

                    }

                    // If Glider is diving
                    if (tiltY > 22 && depth <900) {
                        outPins[EDemoBoard.H0].setLow();
                        outPins[EDemoBoard.H1].setLow();

                        System.out.println("Glider is Diving, ballast off");
                    }
                    if (tiltY > 22 && depth >900) {
                        outPins[EDemoBoard.H0].setHigh();
                        outPins[EDemoBoard.H1].setLow();

                        System.out.println("Reached Max Depth, ballast on to
                        rise");
                    }
                    //  Rolling right
                    if (tiltX < 10) {
                        outPins[EDemoBoard.H0].setLow();
                        outPins[EDemoBoard.H1].setHigh();

                       System.out.println("Glider is rolling  over 10 to
                        right, activate trim ballast.");
                    }

                    // turning left
                    if ( tiltX > 10) {
                        outPins[EDemoBoard.H0].setLow();
                        outPins[EDemoBoard.H1].setHigh();

                       System.out.println("Glider is rolling  over 10 to
                        left, activate trim ballast.");
                    }

                    // Glider is Rising at correct pitch
                    if (tiltY < -22 && depth > 10) {
                        outPins[EDemoBoard.H0].setLow();
                        outPins[EDemoBoard.H1].setLow();
                        System.out.println("Glider is Rising ballast off");
                    }
                    if ( depth < 10) {
                        outPins[EDemoBoard.H0].setHigh();
                        outPins[EDemoBoard.H1].setLow();
                        System.out.println("Glider is at surface, ballast
                        on, begin dive");
                    }



                } catch (IOException e) {
                    //System.out.println("Nothing received");
                }
            }
        }
    }.start();
}

/**************************************************
 * THIS FUNCTION NOT CURRENTLY USED.  IF IN THE EVENT
 * ANOTHER DEVICE IS ADDED TO THE CONTROL SYSTEM
 * CONFIGURATION IT IS READY TO TRANSMIT DATA.
```

```
     *
     * The sender thread sends a string each 500 ms
     ************************************************/
    synchronized public void startSenderThread() {
        new Thread() {

            public void run() {
                // We create a DatagramConnection
                DatagramConnection dgConnection = null;
                Datagram dg = null;
                try {
                    // The Connection is a broadcast so we specify it in the
                       creation string
                    dgConnection = (DatagramConnection)
                Connector.open("radiogram://broadcast:37");
                    // Then, we ask for a datagram with the maximum size allowed
                                    dg =
                dgConnection.newDatagram(dgConnection.getMaximumLength());
                } catch (IOException ex) {
                    System.out.println("Could not open radiogram broadcast
                     connection");
                    ex.printStackTrace();
                    return;
                }

                while (true) {
                    try {
                        // We send the message (UTF encoded)
                        dg.reset();
                        dg.writeDouble(accelerometer.getTiltY());
                        dg.writeDouble(accelerometer.getTiltX());
                        dg.writeUTF("" + accelerometer.getTiltY() + " " +
                        accelerometer.getTiltX());
                        dgConnection.send(dg);
                        System.out.println("Broadcast is going through");
                    } catch (IOException ex) {
                        ex.printStackTrace();
                    }
                    Utils.sleep(500);
                }
            }
        }.start();
    }

    protected void pauseApp() {
    }

    /**
     * Called if the MIDlet is terminated by the system.
     * I.e. if startApp throws any exception other than
     *MIDletStateChangeException,
     * if the isolate running the MIDlet is killed with Isolate.exit(), or
     * if VM.stopVM() is called.
     *
     * It is not called if MIDlet.notifyDestroyed() was called.
     *
     * @param unconditional If true when this method is called, the MIDlet must
     *    cleanup and release all resources. If false the MIDlet may throw
     *    MIDletStateChangeException  to indicate it does not want to be
     *destroyed
     *    at this time.
     */
    protected void destroyApp(boolean unconditional) throws
MIDletStateChangeException {
    }
```

APPENDIX D. RECORD CLASS CODE

```
/**
 *This is the Record ClassSuggested Framework
 * SeaDiver3 Gliding vehicle.  It creates an instance of the statusRecord
 *which
 * will be the container for the current status information.
 *
 * @author Ron Hemmelgarn
 * @version 1.0.0
 *
***********************************************************************
*
*Copyright (c) 1995-2008 held by the author(s).  All rights reserved.

*Redistribution and use in source and binary forms, with or without
*modification, are permitted provided that the following conditions
*are met:

    * Redistributions of source code must retain the above copyright
      notice, this list of conditions and the following disclaimer.
    * Redistributions in binary form must reproduce the above copyright
      notice, this list of conditions and the following disclaimer
      in the documentation and/or other materials provided with the
      distribution.
    * Neither the names of the Naval Postgraduate School (NPS)
      Modeling Virtual Environments and Simulation (MOVES) Institute
      (http://www.nps.edu and http://www.MovesInstitute.org)
      nor the names of its contributors may be used to endorse or
      promote products derived from this software without specific
      prior written permission.

*THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
*"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
*LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
*FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE
*COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
*INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING,
*BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
*LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
*CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
*LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
*ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
*POSSIBILITY OF SUCH DAMAGE.

 ***********************************************************************

       */

      package glider;

      /**
       *
       * @author Ron
       */
      import java.util.Date;
      import com.sun.spot.peripheral.Spot;
      import com.sun.spot.sensorboard.EDemoBoard;

      public class StatusRecord
      {   //VARIABLES
          private String timeStamp;
          private double currentLat;
          private double currentLong;
          private double targetLat;
          private double targetLong;
          private double heading;
          private double depth;
```

```java
private double pitch;
private double rollAngle;
private char rollDirection;
private EDemoBoard eDemo;
    //Constructor
public StatusRecord()
{

}
public double getCurrentLat() {
    return currentLat;
}
public void setCurrentLat(double currentLat) {
    this.currentLat = currentLat;
}
public double getCurrentLong() {
    return currentLong;
}
public void setCurrentLong(double currentLong) {
    this.currentLong = currentLong;
}
public double getDepth() {
    return depth;
}
public void setDepth(double depth) {
    this.depth = depth;
}
public double getHeading() {
    return heading;
}
public void setHeading(double heading) {
    this.heading = heading;
}
public double getPitch() {
    return pitch;
}
public void setPitch(double pitch) {
    this.pitch = pitch;
}
public double getRollAngle() {
    return rollAngle;
}
public void setRollAngle(double rollAngle) {
    this.rollAngle = rollAngle;
}
public char getRollDirection() {
    return rollDirection;
}
public void setRollDirection(char rollDirection) {
    this.rollDirection = rollDirection;
}
public double getTargetLat() {
    return targetLat;
}
public void setTargetLat(double targetLat) {
    this.targetLat = targetLat;
}
public double getTargetLong() {
    return targetLong;
}
public void setTargetLong(double targetLong) {
    this.targetLong = targetLong;
}
public String getTimeStamp() {
    return timeStamp;
}
public void setTimeStamp(String timeStamp) {
    this.timeStamp = timeStamp;
}
```

APPENDIX E. UART LOOPBACK TEST CODE

```
/**
 * @author: Ron Hemmelgarn
 * @date: 19 September
 * @file: seaDiverActController
 *.
 *The purpose of this code is to act as a loopback test for
 * testing the UART ability to send and receive.
 *
 */

package org.nps.auv;


import com.sun.spot.sensorboard.EDemoBoard;
import com.sun.spot.sensorboard.peripheral.ITriColorLED;
import com.sun.spot.util.*;
import java.io.*;
import javax.microedition.midlet.MIDlet;
import javax.microedition.midlet.MIDletStateChangeException;

/**
 * The startApp method of this class is called by the VM to start the
 * application.
 *
 * The manifest specifies this class as MIDlet-1, which means it will
 * be selected for execution.
 */
public class StartApplication extends MIDlet {

    private ITriColorLED [] leds = EDemoBoard.getInstance().getLEDs();
    public EDemoBoard demo;
    protected void startApp() throws MIDletStateChangeException {
        System.out.println("Hello, world");


        while (true) {
            System.out.println(".");
            demo.writeUART((byte) 'T');
            Utils.sleep(1000);
            try {
                while (demo.availableUART() > 0) {
                    try {
                        System.out.print((char) demo.readUART());
                    } catch (IOException IO) {
                        System.out.println(IO.getMessage());
                    }
                }
            } catch (IOException ex) {
                ex.printStackTrace();
            }
        }


    }

    protected void pauseApp() {
        // This is not currently called by the Squawk VM
    }

    /**
     * Called if the MIDlet is terminated by the system.
     * I.e. if startApp throws any exception othe
     *MIDletStateChangeException,
     * if the isolate running the MIDlet is killed with Isolate.exit(),
     or
     * if VM.stopVM() is called.
```

```
     *
     * It is not called if MIDlet.notifyDestroyed() was called.
     *
     * @param unconditional If true when this method is called, the
     *MIDlet must
     *    cleanup and release all resources. If false the MIDlet may
     *throw
     *    MIDletStateChangeException  to indicate it does not want to be
     *destroyed
     *    at this time.
     */
     protected void destroyApp(boolean unconditional) throws
       MIDletStateChangeException {
         for (int i = 0; i < 8; i++) {
             leds[i].setOff();
         }
    }
}
```

APPENDIX F. EXAMPLE OUTPUT DATA FILE

```
/Time:Thu Sep 25 11:26:17 PST 2008/Depth:1024/PITCH:27/ROLL:0/Heading:61/Temp:77.9/Power:5105mv
/Time:Thu Sep 25 11:26:17 PST 2008/Depth:1024/PITCH:25/ROLL:6/Heading:65/Temp:77.9/Power:5105mv
/Time:Thu Sep 25 11:26:18 PST 2008/Depth:1024/PITCH:27/ROLL:1/Heading:63/Temp:77.45/Power:5105mv
/Time:Thu Sep 25 11:26:18 PST 2008/Depth:1024/PITCH:23/ROLL:4/Heading:65/Temp:78.35/Power:5105mv
/Time:Thu Sep 25 11:26:19 PST 2008/Depth:1024/PITCH:22/ROLL:3/Heading:67/Temp:77.9/Power:5105mv
/Time:Thu Sep 25 11:26:19 PST 2008/Depth:1024/PITCH:24/ROLL:2/Heading:65/Temp:77.9/Power:5105mv
/Time:Thu Sep 25 11:26:20 PST 2008/Depth:1/PITCH:22/ROLL:0/Heading:68/Temp:78.35/Power:5105mv
/Time:Thu Sep 25 11:26:20 PST 2008/Depth:1024/PITCH:23/ROLL:4/Heading:66/Temp:77.45/Power:5105mv
/Time:Thu Sep 25 11:26:21 PST 2008/Depth:1024/PITCH:23/ROLL:2/Heading:66/Temp:77.45/Power:5105mv
/Time:Thu Sep 25 11:26:21 PST 2008/Depth:1/PITCH:20/ROLL:-1/Heading:69/Temp:77.9/Power:5105mv
/Time:Thu Sep 25 11:26:22 PST 2008/Depth:1024/PITCH:21/ROLL:2/Heading:68/Temp:77.45/Power:5105mv
/Time:Thu Sep 25 11:26:22 PST 2008/Depth:1/PITCH:22/ROLL:0/Heading:67/Temp:77.45/Power:5105mv
/Time:Thu Sep 25 11:26:23 PST 2008/Depth:1/PITCH:19/ROLL:0/Heading:70/Temp:77.9/Power:5105mv
/Time:Thu Sep 25 11:26:23 PST 2008/Depth:1/PITCH:19/ROLL:-1/Heading:69/Temp:77.9/Power:5105mv
/Time:Thu Sep 25 11:26:24 PST 2008/Depth:1/PITCH:20/ROLL:-1/Heading:69/Temp:78.35/Power:5105mv
/Time:Thu Sep 25 11:26:24 PST 2008/Depth:1/PITCH:18/ROLL:-1/Heading:70/Temp:78.35/Power:5105mv
/Time:Thu Sep 25 11:26:25 PST 2008/Depth:1/PITCH:18/ROLL:0/Heading:71/Temp:78.35/Power:5105mv
/Time:Thu Sep 25 11:26:26 PST 2008/Depth:1/PITCH:19/ROLL:0/Heading:70/Temp:77.9/Power:5105mv
/Time:Thu Sep 25 11:26:26 PST 2008/Depth:1/PITCH:19/ROLL:0/Heading:70/Temp:78.35/Power:5105mv
/Time:Thu Sep 25 11:26:27 PST 2008/Depth:1/PITCH:18/ROLL:0/Heading:71/Temp:78.35/Power:5105mv
/Time:Thu Sep 25 11:26:27 PST 2008/Depth:1/PITCH:19/ROLL:1/Heading:70/Temp:77.45/Power:5105mv
/Time:Thu Sep 25 11:26:28 PST 2008/Depth:4/PITCH:18/ROLL:0/Heading:71/Temp:78.35/Power:5105mv
/Time:Thu Sep 25 11:26:29 PST 2008/Depth:168/PITCH:18/ROLL:0/Heading:71/Temp:77.9/Power:5105mv
/Time:Thu Sep 25 11:26:30 PST 2008/Depth:1024/PITCH:18/ROLL:4/Heading:70/Temp:78.800Power:5105mv
/Time:Thu Sep 25 11:26:30 PST 2008/Depth:1024/PITCH:19/ROLL:2/Heading:70/Temp:78.35/Power:5105mv
/Time:Thu Sep 25 11:26:31 PST 2008/Depth:1024/PITCH:19/ROLL:3/Heading:70/Temp:78.35/Power:5105mv
/Time:Thu Sep 25 11:26:31 PST 2008/Depth:1024/PITCH:19/ROLL:3/Heading:69/Temp:77.9/Power:5105mv
/Time:Thu Sep 25 11:26:32 PST 2008/Depth:1024/PITCH:19/ROLL:3/Heading:70/Temp:78.35/Power:5105mv
/Time:Thu Sep 25 11:26:32 PST 2008/Depth:267/PITCH:17/ROLL:0/Heading:73/Temp:78.35/Power:5105mv
/Time:Thu Sep 25 11:26:33 PST 2008/Depth:141/PITCH:17/ROLL:0/Heading:72/Temp:77.9/Power:5105mv
/Time:Thu Sep 25 11:26:33 PST 2008/Depth:148/PITCH:17/ROLL:0/Heading:72/Temp:78.35/Power:5105mv
/Time:Thu Sep 25 11:26:34 PST 2008/Depth:155/PITCH:17/ROLL:-1/Heading:71/Temp:78.35/Power:5105mv
/Time:Thu Sep 25 11:26:34 PST 2008/Depth:86/PITCH:17/ROLL:0/Heading:72/Temp:78.35/Power:5105mv
/Time:Thu Sep 25 11:26:35 PST 2008/Depth:1/PITCH:17/ROLL:0/Heading:72/Temp:77.9/Power:5105mv
/Time:Thu Sep 25 11:26:35 PST 2008/Depth:1/PITCH:8/ROLL:1/Heading:80/Temp:77.9/Power:5105mv
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX G. PARTS ON HAND

[redacted]

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX H. ADDITIONAL PARTS REQUIRED

ITEM	DESCRIPTION	QTY
Inclinometer	(0-90°) (Center 45°)	3
Accelerometers 3+3		3
Gyros		
Flow Sensors	Piezoelectric sensors- Top and Bottom	2
Planes		2
Batteries	12 Volt 7Ah	?
Battery Controller	Signals: Voltage, Discharge Rate. Fault on Low V.	1
Ballast System Controller		
Main Pump		1
Secondary Pump		1
Ballast	Main and Trim being replaced by dual main	2 Bag
Pressure Tank		1
Check Valves	Check valves, not solenoids.	2
Locator Beacon	Light	1
Hydrophone	Monitor Self Noise	1
CTD	Payload	1

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

Berk, J., & Mitter, N. (2006). *Autonomous Light Air Vessels (ALAVs).* Retrieved September 19, 2008, from Autonomous Light Air Vessels (ALAVs): http://www.alavs.com/ALAVs_ACM_MM2006.pdf

BlueFin Robotics. (n.d.). *BLUEFIN SPRAY GLIDER*. Retrieved March 15, 2008, from BlueFin Robotics: http://www.bluefinrobotics.com/bluefin_glider.htm

Davis, Duane and Don Brutzman, "The Autonomous Unmanned Vehicle Workbench: Mission Planning, Mission Rehearsal, and Mission Replay Tool for Physics-based X3D Visualization," *14th International Symposium on Unmanned Untethered Submersible Technology (UUST)*, Autonomous Undersea Systems Institute (AUSI), Durham New Hampshire, 21-24 August 2005.

Davis, Duane T., *Design, Implementation and Testing of a Common Data Model Supporting Autonomous Vehicle Compatibility and Interoperability*, Ph.D. Dissertation, Naval Postgraduate School, Monterey California, September 2006.

Davis, R., Eriksen, C., & Jones, C. (2003). Autonomous Buoyancy-Driven Underwater Gliders. In G. Griffiths, *Technology and Applications of Underwater Vehicles* (pp. 37-38). London: Taylor & Francis.

Desert Star Systems. (2008). *Desert Star Systems*. Retrieved September 20, 2008, from http://www.desertstar.com/newsite/arc/arc1.html

Dumonteil, R., Gassier, D., & Rebollo, J. (2006). *Implementing a Low-Cost Long-Range Unmanned Underwater Vehicle: The Seadiver Glider.* Monterey, California: Naval Postgraduate School.

Gomez, M. (2001, November 30). *Embedded Systems Design.* Retrieved June 16, 2008, from Embedded: http://www.embedded.com/15201692

Griffiths, G. (2003). *Technology and Applications of Autonomous Underwater Vehicles.* New York, NY: Taylor & Francis.

Jalbert, J., Baker, J., Duchesney, J., Pietryka, P., & Dalton, W. (2003). *Solar-Powered Autonomous Underwater Vehicle Development.* Retrieved September 20, 2008, from AUVSI.org Pulications: http://www.ausi.org/publications/JalbertEtal2003.pdf

Leandri, D. (2008, August 15). Professor, ISITV, Toulon France. (R. Hemmelgarn, Interviewer)

Naval Undersea Warfare Center. (2004). *Navy Unmanned Undersea Vehicle (UUV) Master Plan.* Washington D.C: Department of the Navy.

SeaGlider Summary. (2001, October). Retrieved September 18, 2008, from University of Washington: http://www.apl.washington.edu/projects/seaglider/summary.html

Seguin, J. M. (2007). *Simulating Candidate Missions for a Novel Glider Unmanned Underwater Vehicle.* Monterey, California: Master's Thesis, Naval Postgraduate School.

Sun Microsystems Inc. (2007, October 11). Sun Small Programmable Object Technology (Sun SPOT) Developers Guide. Santa Clara, California, United States of America.

Sun Microsystems Inc. (2007, October 1). Sun Small Programmable Object Technology (Sun SPOT) Owners Manual. Santa Clara, CA, United States of America.

Sun Microsystems Inc. (2007, October). *SunSPOTWorld - Project Sun SPOT Owners Manual.* Retrieved October 30, 2007, from SunSPOTWorld - Home of project SunSPOT: https://www.sunspotworld.com/docs/Purple/SunSPOT-OwnersManual.pdf

Sun Microsystems. (2008, January 29). *Sun News*. Retrieved July 23, 2008, from http://www.sun.com/aboutsun/pr/2008-01/sunflash.20080129.3.xml

Systronics. (n.d.). *TrackBot Home @ Systronix*. Retrieved September 20, 2008, from TrackBot @ Systronix: http://www.trackbot.systronix.com/

University of Washington. (2006). *XRay Flying Wing Glider Summary*. Retrieved September 20, 2008, from University of Washington Applied Physics Laboratory: http://www.apl.washington.edu/projects/xray/summary.html

USGlobalSat. (n.d.). *ET-202 User Manual.PDF.* Retrieved December 5, 2007, from SparkFun.com: http://www.sparkfun.com/datasheets/GPS/ET-202%20User%20Manual.pdf

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Don Brutzman
   Naval Postgraduate School
   Monterey, California

4. Jeff Weekley
   Naval Postgraduate School
   Monterey, California

5. Doug Horner
   Naval Postgraduate School
   Monterey, California

6. Dan Boger
   Naval Postgraduate School
   Monterey, California

7. Ray Jones
   Naval Postgraduate School
   Monterey, California

8. Tony Healy
   Naval Postgraduate School
   Monterey, California

9. Didier Leandri
   University of Toulon
   Toulon, France

10. Marco Flagg
    Desert Star Systems
    Marina, California

11. Tom Swean
    Office of Naval Research
    Arlington, Virginia

12. Pierre Corriveau NPRI
CTO, Naval Undersea Warfare Center
Newport, Rhode Island

13. D. Richard Blidberg
Autonomous Undersea Systems Institute
Lee, New Hampshire

14. Dr. Bill Smuda
Army Research, Development, and Engineering Command (REDCOM)
Aberdeen Proving Ground, Maryland

15. John Moore
Navy Modeling and Simulations Office
Ft. Belvoir, Maryland

16. Rick Goldberg
Aniviza Inc.
Los Gatos, California

17. Alan Hudson
Yumatech, Inc.
Seattle, Washington

18. Distinguished Professor Anthony Healy
Department of Mechanical Engineering
Naval Postgraduate School
Monterey, California

19. Sean Krageland
Department of Mechanical Engineering
Naval Postgraduate School
Monterey, California

20. Dr. Thomas Curtin
Office of Naval Research
Arlington, Virginia

21. Steven Chappell
Autonomous Undersea Systems Institute
Durham, New Hampshire

22. Rick Komerska
Autonomous Undersea Systems Institute
Durham, New Hampshire

23. Peter Flynn
Naval Research Laboratory,
Stennis Space Center, Mississippi

24. Mark Falagh
L-3 Communications
Orlando, FL

25. Eyton Pollach
L-3 Communications
Orlando, FL

26. Dr. Mikhail Auguston
Naval Postgraduate School
Monterey, California

27. Eyton Pollach
L-3 Communications
Orlando, Florida

28. Eric Chaum
NUWC
Newport, Rhode Island

29. David Bellino
NUWC
Newport, Rhode Island

30. CAPT Jeff Kline, USN (Ret.)
Operations and Research Department
Naval Postgraduate School
Monterey, California

31. Duane Davis
Naval Postgraduate School
Monterey, California

1. Nick Polys
Virginia Tech
Blacksburg, Virginia